Documentation for the
Marine Jurisdictions Database

Roger A. Goldsmith
Woods Hole Oceanographic Institution
June, 1998

19990309040

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

DTIC QUALITY INSPECTED 1

Documentation for the Marine Jurisdictions Database

Roger A. Goldsmith
Woods Hole Oceanographic Institution

June, 1998

Contents

Disclaimer

Maps and charts used to come with all sorts of disclaimers, cautions and warnings. Phrases like 'not suitable for navigation' 'subject to change' and perhaps most appropriate 'the prudent mariner will not rely solely on any single aid to navigation,...' used to adorn every chart but for whatever reason are not as common now. There should be no such illusions for this product. The boundaries are the best representations that could be constructed at this time for this project. They will undoubtedly change and can be improved as more information and resources become available. In particular, there are a number of areas where claims are not recognized by one of the parties involved, or by third party states of the international community. Assignment of points and boundaries to specific countries are meant to be advisory or for example only and do not represent advocacy or primacy of any claim. Recognition of any claims in this database is neither expressed nor implied and it is the user's responsibility to verify the current claims status with the Department of State (USA), coastal states and other sources.

Copyright © Roger Goldsmith, Woods Hole Oceanographic Institution, 1998
All rights reserved.

The information in this document is subject to change without notice.

The names of companies and products herein are trademarks or registered trademarks of their respective trademark owners.

1.0 Background

Researchers and other oceanographic mission planners are often confronted with cruises that approach or enter territorial waters claimed by foreign countries under a variety of jurisdictional agreements or proclamations. Work in these waters usually requires some type of notice, clearance or authorization. New claims are being made even as old claims are still being negotiated between governments or presented before international bodies such as the Convention on the Law of the Sea or the International Court of Justice.

A chart [Ross, 1992] delineating the various established boundaries then in effect provided some guidance as to the extent of territorial water (and the corresponding reduction of 'open-seas'). The chart was well received by the oceanographic community and there have been ongoing requests for not only the charts but also the digitized boundaries.

There were, however, several problems were faced in the generation of 'Maritime Claims' chart. There appeared to be no single repository for all the various types of jurisdictions. While the U. S. Department of State was of considerable help, information was well distributed and buried in several offices. Nor could they offer much information about claims being adjudged between countries not involving the United States. Once these claims became 'official' there seemed to be no mechanism for dissemination to third parties or any central repository. Finally, while the chart was prepared using digital representations of the boundaries, they were organized only in the very coarsest of categories for the jurisdictional elements being portrayed. There was no topology, that is the ability to associate properties, such as country, with any point or line. Thus, while the digital boundaries were distributed to a variety of requestors, they have all been used simply to regenerate the chart boundaries using a variety of software packages.

The purpose of this project was to take the data gathered for the Maritime Claims chart and create a Maritime Jurisdictions digital database suitable for use with oceanographic mission planning objectives. To accomplish this a themed data set, associated tables, prototype processing software and documentation was developed. While GIS systems readily lend themselves to this task and the eventual application of the resulting data set, this product is generated as ASCII files with positions recorded in decimal degrees. This was done so as to maximize the deployment options. Users should be able to incorporate this data set into their own applications using a variety of hardware and operating systems. In this context it should also be noted this is not a deployable navigation and display application. The operating environments of the potential user community are simply too diverse. For display applications alone there are software packages such as Matlab, gmt, Surfer/MapViewer, ArcView, Atlas GIS, AutoCAD and many others, all driving a variety of CRTs, pen plotters and laser printers. To accommodate this diversity the product was made as a simple, easy to understand database that can be incorporated into a variety of applications.

2.0 Project directory structure

Specific directory path names are of course platform dependent and for that reason the directory structures have been kept fairly simple. The various subdirectories were determined primarily with functionality and ease of maintenance in mind. The software utilities, discussed in later sections, usually allow the user to specify the paths and specific file identifiers. Naming conventions of both directories and files has adhered to the 8.3 convention.

2.1 Parent directory MJDB

The parent directory is identified as MJDB and may be placed at the top level or under some other directory in the user's directory structure. There are no implicit restrictions on the paths or even the path identifiers for that matter. Some of the utility software developed in conjunction with the boundary data allows for the use of default paths and file identifiers but these may me easily set or overridden by the user.

The MJDB directory contains, mostly as a matter of convenience, some of the data definition files (MJ_CNTRY, MJ_AREA and MJ_ATTR) as well as the computed descriptive files (MJ_LRNG, MJ_LINED, MJ_ARNG, MJ_BRNG and MJ_LINEB). These are discussed in more detail in later sections.

2.2 Subdirectory SEGS

The subdirectory SEGS is the repository for all the files that comprise the boundaries of the marine jurisdictions. The files are named using the convention

ls****.seg,

where the '****' field represents a four character integer in the range 0001 to 9999. In practice the range has been restricted from 0001 to 0999; note the field must be leading zero-filled. The use of some combination of country codes for the naming convention was considered but any advantages accrued was out-weighted by the complexity introduced with multiple country masses (Kiribati) or contacts (United States and Canada). While the use of numbers makes the files a little less readily identifiable, each file is internally self-documenting and the definitions file (MJ_LINED) allows a quick search of various topological properties associated with the line segment. The contents of the segment files are described in more detail in a later section.

In theory the segment identifiers may range up to 9999. In fact, to import the data to a true GIS system there would probably not be any problems. In the support software developed in association with this project there are some explicit limits, such as in the array sizes, and some implicit limits. The latter involves issues of storage methods and access by direct indexing or table lookups. For this initial project the segment identifiers are used as direct indices to the storage arrays. This is a workable arrangement as long as the arrays are neither too large nor sparse. If segment identifiers covering the whole range 0 to 9999 are desired the software design should be revisited.

2.3 Subdirectory BASE

The subdirectory BASE contains the files of base point data. A few words of clarification are necessary here. The term 'baseline' seems to be used in at least a couple of different contexts in the literature and general discussion. In some cases it has been applied to segments that comprise the actual country boundaries. This usually seems to occur in cases where the boundary is an extension of the land boundary between neighboring countries. In the second context the term refers to a series of line segments which comprise outermost edge of enclosed waters. At this point the application of the baseline concept gets a bit murky and political. The line may simply run along land's edge but this introduces the requirement for defining the term 'land'. Is it high tide, low tide, mean tide? Does it include reefs or rocks that are sometimes submerged? The concept is further complicated when a 'baseline' connects two headlands, with resulting bay now being considered interior to the polygon which encloses the 'land mass'. Finally, regardless of what a country may claim as the relevant definitions, it all comes down to whether or not other countries or the body of world opinion decides to accept the definition.

In this project I have taken the latter definition of 'baseline' and reduced it even further to that of 'base point'. There were two main reasons for this decision. First, as described above, there was no consistency in the reporting of baseline data. Indeed, far less than half the countries have the information readily available. Beyond the attendant problems of definition there were also a variety of methods used to describe the baselines. Some countries reported the points of a continuous enclosing polygon. Others reported base point pairs, sometimes contiguous, sometimes not, it seemingly being left to the reader to figure out what happens in the gaps. Sometimes the document said that an arc was to be drawn in the gap.

The base points, once established and agreed by all parties, do not objectively determine the resulting country boundary. This was the second and most compelling reason for including only the base point positions and not the purported baselines. For a small isolated island it may help in determining the foci of the 200 nm. radii to the limit. In most cases however, there are neighboring countries with conflicting claims to be considered. Then the base points become the jumping-off basis for protracted subjective decisions about where the boundary actually lays. In other words, given the baselines, or base points, one still would not be able to determine where the boundary was located without having a treaty or some additional documentation in hand.

The result of all this means the base points, where available, have been stored in files in the BASE subdirectory but they are pretty much used only as a reference. They might help show where claims are in conflict or where a boundary has been defined or entered erroneously but do not expect much beyond that. The files are named using the convention

```
bl****.bas
```

where the "****" field represents a four character integer in the range 0001 to 9999. In practice the range has been restricted from 0001 to 0999; note the field must be leading zero-filled. The contents of the files are described in more detail in a later section.

2.4 Subdirectory DOC

The DOC subdirectory contains this document and related information such as figures. The document is stored in common formats such as Microsoft Word 97 and ASCII text. Figures are stored in PostScript.

2.5 Subdirectory MFILES

Much of the initial processing effort for this project was done using the Matlab software. The processing platform, an old DECstation, had an old version (4.2c) of the package and I would not recommend doing any prolonged processing in this manner. Newer releases of Matlab (version 5.0 and higher) contain a Mapping toolbox and with the speed of a new PC, or workstation, this combination provides an interesting, low cost method of developing some basic mapping and charting applications. I have included some of the m-files used in processing the MJDB data in this directory. They can be used in the with the basic Matlab package; in fact there are no references to the Mapping toolbox so the user will have to interface the utilities with their application. The functions are meant primarily to assist with the task of extracting various elements of the MJDB. Feel free to rewrite them as necessary. As I said, they were done with an old version of Matlab and with the idea of providing a simple prototype to help users understand what the heck is going on. The various modules are described in more detail in a later section and also are documented internally in a format compatible with the Matlab HELP facility.

2.6 Subdirectory SRC

While the data files comprising the boundary defining line segments are the primary products of this project, it was necessary to have some way to check the processing and simulate the applications in which they might be incorporated. Not everybody has a full-fledged geographic information system (GIS) or software packages like Matlab or Surfer. In fact it was expected that applications already existed that would simply embed the new jurisdictional limits functionality. Toward that end a suite of programs was developed which would hopefully assist the applications developer with the task of extracting various elements of the MJDB.

Several factors were considered during the writing of these programs. All programs were written using the C language as this (or C++) currently seems to be the most widely used language among newer applications. Programs written in FORTRAN should not have much trouble interfacing to these utilities. There is undoubtedly room for sophisticated use of structures and lots of other neat programming tricks here. The intent was to keep things simple enough for programmers with limited experience to understand and incorporate the MJDB into their own application. To that end it was assumed that reasonable amounts of memory and adequate processing speed would be available to the user. State-of -the-art compilers might not be so do not expect anything out of the ordinary in programming techniques here. Even ANSI compiler options were something not readily available in the development process. With that said, the programs compiled without error on both a DECstation using the Ultrix C compiler and on a Windows NT system using the Watcom C/C++ (Version 11) compiler. Note I said 'no errors' and not 'no warnings'. Different compilers have different tastes and there seems no accounting for the latter.

Also included in this subdirectory are the appropriate Unix 'makefiles'. For those who like to know what is going on I have included the 'makefile' for each program. These take the form

```
<program>.mak
```

and illustrate which types of applications required which modules. For those who like to push one button and plunge fearlessly forward there is the all encompassing

```
Makefile
```

Which allows the compilation and executable building of individual applications or everything. This form also constructs a library ('libmjdb.a') of all the utility routines. The PC applications used the Watcom target management system to build the executables again using an assembled library of the utility modules. The PC executables have been included in the PKZIP version of the directory, as they generally seem to work under most version of DOC. Again, for portability considerations, the programs are designed to be run under DOS rather than be dependent on any type of Windows or NT drivers.

Almost all the software modules adopt the convention of beginning with the character string 'mj'. The programs fall into three main categories. These are

1. Utility modules, including the include file ('mj_incl.h'), of the form 'mj_***.c'
2. Test modules, of the form mjt_**.c

3. Applications prototypes, of the form mja_***.c

Each of these is discussed in more detail in later sections and is documented internally.

3.0 Components

The primary product of this project is the data set of line segments delineating countries jurisdictional limits. To facilitate the use of these segments several auxiliary files are also included. These include summaries of the country codes, identifiers, line segment ranges, area ranges and boundary definitions. Information has also been included for the points defining country baselines. Finally, a suite of software has been included which may be used to test the product and assist the user in the development of applications software. Each of these components is discussed in more detail in the sections that follow.

3.1 Line segment files

The main component is the line segment files that make up the maritime boundaries. In the strictest sense it is the coordinates that make up the line segments that determine the boundaries. By that definition each coordinate should be identified as to its source and which line set(s) it belongs in. This project does not go to that extreme. The line segment files are the base unit and these files contain the information needed to construct the maritime boundaries of the selected areas. The information was taken from the digital data used to construct the Maritime Claims chart [Ross/Fenwick]. The principle effort of this project was the processing of that data set to obtain a topologically consistent network of line segments defining the maritime boundaries of the world's nations. Those files were then updated with the information furnished by the United States Navy's Office of Naval Research in the "Maritime Claims Reference Manual". These files are stored in the SEGS subdirectory and are identified by the following file naming convention:

```
ls****.seg
```

In an effort to make this data set as portable as possible these files are stored in ASCII format. At first glance this might seem to make the file sizes needlessly large. In fact, storing a {longitude latitude} coordinate pair as machine dependent binary values would require 8 bytes in single precision, 16 bytes for double precision. Storing the data for a 100-meter resolution (about 0.1 minutes) requires that decimal degrees be stored to at least a 0.001 resolution. The data here has been stored to the 0.000001 resolution, though that should not be construed as the accuracy of the coordinate positions. In terms of storage space though, this requires only 22 bytes per coordinate. It was felt the clarity of the ASCII format, for portability and on-going maintenance, outweighed the manageable increase in size.

A more significant factor in determining the file size is the number of points used to define a segment. Where the boundaries are defined by treaty or other landmarks this number is usually small, on the order of ten points. The original data set obtained for the Maritime Claims chart was apparently generated by a digitization process from some unknown source. The lines were often digitized at a resolution of about one-kilometer resulting in segments containing on the order of hundreds of points. In most cases no attempt was made to decimate the segments and the data has been passed along in its original form.

The use of a numbering scheme in the file naming convention does not immediately lend itself to a quick visual recognition of the data contained therein. Each file was therefore designed to contain both comment records and a descriptive header record. The comment records allow for a more descriptive identification of the data and also provide a mechanism for identifying the source of the data. The descriptive header provides the topology of the line segment as well as some description of the contents of the file. A complete description of the file format and record content is provided in Appendix A. The descriptive header information is also used to construct the line definition file (MJ_LINED) which is used to facilitate some of the routine processing and analysis.

Ideally, only one segment would be necessary to represent the border between two neighboring areas. In practice however, several segments are sometimes required to adequately define an interface. A large segment containing over 1000 coordinates was sometimes broken into two or more smaller segments. In some cases segments approached or diverged from a common point in opposite directions. Rather than reordering and combining, the segments were just given separate identification numbers. Finally, as new information became available whole segments, multiple segments or portions of the segments were replaced. The later case requires note about the source. As indicated, almost all the segments were originally obtained by processing the Maritime Claims data set, source 001. As updates were applied the closely spaced one-kilometer digitized data could often be replace by just a few exactly located coordinates. If it was just a portion of the line two options were available: 1) create one or two new segments in addition to the updated segment, or 2) update the source identification code. In these circumstances I generally chose the latter as implicitly everything is from the original source. Two features may help identify these situations. First, specified coordinates, whether by treaty or otherwise, are generally discreet in the sense they are usually only defined to the nearest 0.1 minutes. Additionally, these coordinates are usually defined in a treaty or list of some sort along with a reference number of letter. This reference identifier, usually the position in the list, has been carried along in the segment file as a third column.

The identifiers associated with the various line segments do not have any strict numbering scheme. That said, there are some informal conventions. Segments 500 through 899 are usually associated with borders to the open sea. Thus, most islands may be found in this range. Those segments beginning at 951 are land boundaries but in al other respects follow the format and content conventions. Why are there land boundaries you wonder? The segments that constitute the enclosing polygon border usually start and end at a land-sea interface, though not for islands of course. A straight line connecting the end-points implicitly closes the polygon. Is this correct? The answer is no. However, the next question is 'How do we close the polygon?' Should it be the country land borders? That would allow some nice total area computations and a complete picture of the national extent. But this would introduce a lot more data that seems to be just as unsettled as the maritime limits. Should it be the coastline? It would give a nice picture of the maritime extent but would also introduce even more data and leave one wondering about high tide, low tide, the baseline issue and offshore islands. The most open-ended solution was to let the user application determine how the polygons are closed for anything beyond the default case. That was the argument but there were some cases that were just so egregious that they had to be dealt with, if not for appearance sake then for the point-in-polygon tests that is sort of the whole purpose of this project. In these cases enough of the relevant land border was extracted and used to ensure a more suitable closure. In general the borders were heavily decimated and if the user wishes to implement their own polygon closure method they may do away with these first-pass solutions.

3.2 Area (country) code definition file, MJ_CNTRY

The line segments make use of area identifiers to determine the topology of the maritime borders. The ISO-3166 conventions were adopted as the two-character identifiers used for country/territory identification. These have been linked with a full country name and a numeric identifier in the file MJ_CNTRY. The two-character codes are useful for a quick association of the area in question and are found in most of the associated files. However, when it came time to specify area identifications in associated files a numeric identifier. Many of the arguments used in the choosing of line segment identifiers applied here. Furthermore, there are several instances where countries have multiple, separated components or distant, semi-autonomous territories. It is possible to set up the topology for these cases, as was done for some of the simpler cases, it seemed easier to treat these regions as separate areas. If a new two-character code was assigned here a conflict might arise later when a new code is assigned. The use of the numeric code as the basis for any association, and relegating the two-character code to a descriptive status, will hopefully forestall some of these problems.

The numeric values used have some basis in precedent in that they were used for the WOCE experiment. At one time they were based on the ATT country codes but beyond that I am not sure were they came from. It really doesn't matter as long as they are unique. Using the numeric convention also allows the identification of non-country areas such as the various oceans. The original codes were in the range 0001 to 999; codes grater than 999 were developed for this project. Users who wish to change codes need to make sure the corresponding codes are changed in the area definition file (MJ_AREA) and the segment files. (Changes to the later files should be accompanied by updating the MJ_LINED file.)

The entries in the MJ_CNTRY file are initially sorted by the two-character code but this is not a requirement. The file may be sorted in any order, or not sorted at all. Most of the applications developed load this file at initialization time. Keep in mind that applications should not hard-wire locations in the table in case the order is changed or a subset of the table is used. A more detailed description of the table format and content is given in Appendix B.

3.3 Line segment range file, MJ_LRNG

This file is a secondary product derived from the line segment files. The processing is performed by the support utility routine 'mj_clr', described in another section of this report. The geographic minimum and maximum extents of the line segment are determined and the result is put in this file for each line segment. All range values are converted to positive degrees. Some of the segments will have a maximum longitude greater than 360 degrees as a result of straddling the Greenwich Meridian. A more detailed description of the file format and content may be found in Appendix C.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. By pre-computing the line segment ranges it speeds up operations like the computation of the area ranges. There is no order requirement but the manner in which the ranges are computed results in a numerical line segment order. There is also no limit on the size of this file. Having said that however, there are some limits associated with the number and maximum value of segment identifiers in some of the prototype software.

3.4 Line segment definition file, MJ_LINED

This file is another secondary product derived from the line segment files. The support utility routine 'mj_clr', described in another section of this report, generates the file MJ_LINED' at the same time as the 'MJ_LRNG' file. The contents of this file are basically the same as appears on the line descriptor record of the line segment files. A more detailed description of the file format and contents is contained in Appendix D. The topology information is used to speed operations such as the search for segments belonging to an area. In cases such as disputed borders an area may have associated line segments that do not appear as part of the defined border. This type of topology, as well as line type and other information, is contained in the line segment definition file.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. There is no order requirement but the manner in which the segments are processed results in a numerical line segment order. Again, there is no limit on the size of this file, though there may be some limits associated with the number and maximum value of segment identifiers in some of the associated software.

3.5 Area (country) border definition file, MJ_AREA

The border definition file 'MJ_AREA' is used to determine the line segments that comprise the marine jurisdictional limit of an area or country. Furthermore, the order and sense of the line segments is specified so as to construct a polygon. The result is a continuous line enclosing the selected area to the right-hand side as one moves along the coordinates of the line. (As described earlier, the polygon may be implicitly closed across the land surfaces of the area.) A complete description of the file format and content is given in Appendix E.

The are several ways to specify the topology of an area. Most involve some sort of linked list and that is essentially what this file is. Strictly speaking only the order of the line segments is needed, with the sense (area to the left or right) being determined from the line segment descriptor record or the descriptor file 'MJ_LINED'. I have adopted the additional convention of signing the numerical line segment identifier as a way of expediting the processing for the utility software. It also makes the file more intuitive when checking for consistency.

Only countries with maritime boundaries are included in the initial version of the file. There are some areas bordering territorial seas such as the Baltic and Mediterranean that have not been completely defined. Theoretically there can be any number of areas in the file and each area can contain as many segments identifiers as necessary to describe the boundary. The processing software generally has some limits however.

3.6 Area (country) range file, MJ_ARNG

This file is a secondary product derived from the area definition and line segment range files. The processing is performed by the support utility routine 'mj_car', described in another section of this report. The geographic minimum and maximum extents of the area are determined and the result is put in this file for each area defined in the 'MJ_AREA'. All range values are converted to positive degrees. Some of the segments will have a maximum longitude greater than 360 degrees as a result of straddling the Greenwich Meridian. A more detailed description of the file format and content may be found in Appendix F.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. By pre-computing the area ranges it speeds up operations like the point-in-polygon searches. There is no order requirement but the manner in which the 'MJ_AREA' file is processed implies this file will be in the same order.

4.0 Other components

Several other components of the data are provided. None of these are necessary for the implementation of the Marine Jurisdiction database. They are a combination of useful information and data sets that were helpful in the processing stage of the project. Information such as the points determining the baselines might also be necessary if the project evolves into a more sophisticated GIS system.

4.1 Base points

As detailed in an earlier section, the baseline data is not very complete; documents or data were available for fewer than 25% of the countries. And in general the maritime limits cannot be determined from the baseline data alone. However, as the data was available and being processed at the same time as the boundary data it has been carried along here in an analogous format. The two types of derived files, the ranges and the descriptions are also provided.

4.1.1 Base point files, bl****.bas

The coordinates of the points comprising the baselines are stored in the 'BASE' subdirectory. The file naming convention was described earlier. As not much that can be done with the current state of the data, the points were entered in a format similar to that used for the line segments. A more detailed description is given in Appendix G. The primary difference is the points should be treated individually. While in some instances connecting them may form a contiguous 'shoreline' that is generally not the case. Various documents describe the points as being connected sequentially or in pairs, by straight lines or arcs. Ideally the baseline points along the coast would be combined with the points defining the land borders and thereby create the defining territorial boundary. That effort was beyond the scope of this project but perhaps the baseline data gathered here will provide the impetus for further work in the area.

Other subtle differences are evident in the use of the file format. Because the baselines are internal to the country there is no associated area topology. The fields are here left intact for compatibility with the boundary line segment format but are both filled with the same area identifier value. The line-type and attribute fields similarly contain only generic values.

4.1.2 Base point range file, MJ_BRNG

The baseline range file and the following descriptor file are generated primarily to provide symmetry with the boundary line segment files. It provides a good initial check of the values. Ultimately, if the land borders were included for a country this file might be a useful step in computing the maximum extent of the territorial boundaries in much the same manner as the 'MJ_LRNG' file is used in the creation of the 'MJ_ARNG' file. But for now the file is not really used for anything. The format, analogous to that used in the 'MJ_LRNG' file, is described in Appendix H.

This file is a secondary product derived from the base-point files. The processing is performed by the support utility routine 'mj_cbr', described in another section of this report. The geographic minimum and maximum extents of the base points are determined and the result is put in this file for each baseline. All range values are converted to positive degrees. Some of the baselines may have a maximum longitude greater than 360 degrees as a result of straddling the Greenwich Meridian.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. There is no order requirement but the manner in which the ranges are computed results in a numerical order. There is also no limit on the size of this file. Having said that however, there are some limits associated with the number and maximum value of segment identifiers in some of the prototype software.

4.1.3 Base point definition file, MJ_LINEB

This file is another secondary product derived from the base point files. The support utility routine 'mj_cbr', described in another section of this report, generates the file MJ_LINEB' at the same time as the 'MJ_BRNG' file. The contents of this file are basically the same as appears on the descriptor record of the base-point files. A more detailed description of the file format and contents is contained in Appendix I. The topology information is used to speed operations such as the search for segments belonging to an area.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. There is no order requirement but the manner in which the segments are processed results in a numerical base-point order. Again, there is no limit on the size of this file, though there may be some limits associated with the number and maximum value of baseline identifiers in some of the associated software.

4.2 Attribute file, MJ_ATTR

The attribute file was developed as a place to store various attributes that might be associated with an area. One of the first applications was the need to identify groups of countries, such as all those that had approved baselines, ratified the Law of the Sea treaty or some other qualifying attribute. Of course this is all very straightforward with a database and GIS system. Some of the properties include in the area border definition file might also have been more readily incorporated in this file.

The file is not required as part of the Marine Jurisdictions Database. It is used only in conjunction with the support software provided as part of this project. There is no order requirement but if the entries are not the same as defined in the MJ_AREA file the user will have to implement a method of indirect indexing. The preliminary version of this file has arbitrarily assigned the color of white [1.0 1.0 1.0] to all countries. Some oceans have also been inserted and given the color [0.0 1.0 1.0]. The user may of course develop one or many attribute tables implementing any color scheme they choose.

4.3 Location file, MJ_LOC

Another potentially useful data set contains the coordinates of locations associated with the marine boundaries. There are many sources for this type of data. The locations of ports for many countries can be obtained from the National Imagery and Mapping Agency (NIMA). The ports, along with country capitals, have been placed in the file 'MJ_LOC'. The purpose here is not to duplicate the efforts of NIMA but to provide some convenient utility to this initial project and give some examples of what might be pursued in the future. The World Port Index developed at NIMA is an extensive compilation of ports and associated facilities and should be reviewed.

The version of the file developed here includes both the numeric and two-character country codes to assist in the identification and maintenance of the locations. Other types of point information could also be placed in this or similar files. Here, the file also contains information for countries that do not have marine borders. A more sophisticated version of the data set might also wish to keep track of the type of the location but that is beyond the scope of this project. There are no order requirements for this file, though in this version the entries have been sorted by the two-character country code and by city name within the country. A more detailed description of the file is contained in Appendix K.

5.0 Support software

The software developed during the processing of the boundary data has been included to serve as both documentation of the data structure and provide some example of its use. The Matlab routines were used primarily for the processing and a few enhancements were added to provide some applications utility as well. These modules are stored in the 'MFILES' subdirectory.

The C language programs were developed as tests to ensure the database integrity and suitability for applications development. Additional modules were developed to provide the same functionality as the Matlab routines. These have all been placed in the 'SRC' subdirectory along with associated Unix-style 'makefiles'.

5.1 Matlab routines

For those unfamiliar with Matlab, it is a relatively inexpensive 'numeric computation and visualization software' package that is widely used on both Unix and PC platforms. Its command language provides a large library of routines that allow users to perform interactive analysis or construct their own applications. A number of toolboxes are provided that extends the applications into a variety of fields such as signal processing and mapping. The Matlab routines presented here were developed using the basic Application Toolbox but are compatible with the Mapping Toolbox. All the functions have the conventional comments at the beginning that work with the Matlab 'help' facility. A summary of the routines is given in the 'Content.m' file and is shown in Appendix L. A slightly more detailed description is given in the sections that follow but the best documentation is contained in the code itself.

In the following sections the routines are broken into initialization and retrieval classes. In a rough manner they also are the order in which they might be invoked. A final section is presented with some sample applications.

5.1.1 mj_init.m

The routine 'mj_init' is the initialization and should be the first one called. It sets up the default file locations and identifiers. A function that checks the type of machine does this. The user should edit these routines for use in their environment. Finally, it makes available to the user a number of global variables containing information such as the area identifiers, line and area ranges and area attributes.

```
Usage:        mj_init
```

5.1.2 mj_iac.m

The routine 'mj_iac' initializes the area identifiers by loading the arrays CN, C2, C3, and CT. It processes the area code definition file ('MJ_CNTRY') by default but allows the user to load a file of their choosing. This initialization step is almost always necessary.

```
Usage:  mj_iac (mj_cntry_file)
  where
      mj_cntry_file  is the optional file identifier specifying the location of
            the file.
Returns:  The global arrays CN, C2, C3 and CT are stored.
```

5.1.3 mj_ilr.m

The routine 'mj_ilr' initializes the longitude and latitude range arrays associated with each line segment. This initialization routine is necessary before a variety of subsequent routines.

Usage: mj_iar (r_file)
  where
    r_file   is the string identifying the location of the line range file ('MJ_LRNG'). The 'MJ_LRNG' file is created during execution of the program 'mj_clr.c'. The file simply contains the {lon lat} ranges associated with each line segment to simplify searches.

Returns: The global array LRANGE is stored. It contains the following columns: line_code, lon_min, lon_max, lat_min, lat_max

5.1.4 mj_iar.m

The routine 'mj_iar' computes the geographic range of an area based on the range segments comprising the area. This initialization is necessary before using the point in area routines. This module does not load the pre-computed 'MJ_ARNG' file but rather uses the line segment range file 'MJ_LRNG' and the area definition file 'MJ_AREA' to compute the ranges for the current application.

Usage: mj_iar (a_file, r_file)
  where
    a_file is the string identifying the location of the area definition file. The default is the file 'MJ_AREA'.
    r_file is the string identifying the location of the line segment range file. The default file 'MJ_LRNG' could have been created by running the program 'mj_clr.c'.

Returns: The global array ARANGE, containing values of the area code, longitude minimum, longitude maximum, latitude minimum and latitude maximum, is computed.

5.1.5 mj_iaa.m

The routine 'mj_iaa' loads the area attributes from a file to the internal arrays.

Usage: mj_iaa (a_file)
  where
    a_file is the string identifying the location of the area attribute file. The default is the file 'MJ_ATTR'.

Returns: The current version of the routine loads the color array 'COL_RGB'.

5.1.6 mj_gcn.m

The routine 'mj_gcn' requires a user specified two-character area code and returns the area identifier. The basic utilities are in the routine 'mj_gac'; this is just a shell.

Usage: [area_n] = mj_gcn (area_c2)
  Where
    area_c2   is the two-character area identification code.

Returns: The numeric area identifier or zero if none is found.

5.1.7 mj_gc2.m

The routine 'mj_gc2' requires the user to specify a numeric area identifier and returns the two-character area code. The basic utilities are in the routine 'mj_gac'; this is just a shell.

Usage: [area_c2] = mj_gc2 (area_n)
  where
    area_n   is the integer area identification code.

Returns: The two-character area code or null if none is found.

5.1.8 mj_gc3.m

The routine 'mj_gc3' requires the user to specify a numeric area identifier and returns the three-character area code. The basic utilities are in the routine 'mj_gac'; this is just a shell.

```
Usage:  [area_c3] = mj_gc2 (area_n)
  where
      area_n            is the integer area identification code.
Returns:  The three-character area code or null if none is found.
```

5.1.9 mj_gct.m

The routine 'mj_gct' requires the user to specify a numeric area identifier and returns the associated area descriptive text. The basic utilities are in the routine 'mj_gac'; this is just a shell.

```
Usage:  [area_str] = mj_gct (area_n)
  where
      area_n            is the integer area identification code.
Returns:  The text string array containing the full text identifier of the area
      or null if none is found.
```

5.1.10 mj_gac.m

This routine is the basic area code transformation utility. Given an area code in one form it returns the code in the requested form.

```
Usage:  [area_id, area_str] = mj_gac (method, code_n, code_a)
  where
      method is the integer value specifying the type of request;
                  -3 - given 3 character code, find numeric code;
                  -2 - given 2 character code, find numeric code;
                   2 - given numeric code, find 2 character code;
                   3 - given numeric code, find 3 character code;
                   4 - given numeric code, find text identifier.
      code_n is the numeric country code.  This is ignored if the method is
            negative; any value may be given.
      code_a is the alphanumeric country code.  This is ignored if the method is
            positive; any value may be given.
Result:  The routine returns both the numeric and string identifiers for the
      area or zero and null if the area was not found.
```

5.1.11 mj_gal.m

The routine 'mj_gal' retrieves the identifiers of all line segments bounding the user specified target area. It does this by searching the user specified line definition file.

```
Usage: [seg_id] = mj_gal (d_file, area_n)
  where
      d_file is the string identifying the location of the line descriptor
            format file.  The default is the 'MJ_LINED' file created by the
            program 'mj_clr.c'.
      area_id is the numeric area code identifier of the target area.
Returns:  The vector containing the qualifying line segment identifiers.  The
      lines are retrieved in the order encountered.  Negative identifiers
      indicate reverse order of the line using the convention of target area on
      the right.
```

5.1.12 mj_grl.m

The routine 'mj_grl' is used to retrieve the line segment identifiers of all line segments whose range overlaps a user specified rectangular geographic range. To perform this analysis it uses the range array 'LRANGE' loaded by the routine 'mj_ilr'.

```
Usage:  [seg_id] = mj_grl ([range])
  where
      range is a vector of the form [lon_min  lon_max  lat_min  lat_max] used to
            specify the 'rectangular' geographic region of interest.
Returns: The vector containing the qualifying line segment identifiers.  The
      lines are retrieved in the order encountered in the LRANGE array.  Note
      that while the range of the line may overlap the specified area the line
      segment itself may fall entirely outside the area.
```

5.1.13 mj_gapl.m

The routine 'mj_gapl' is used to retrieve the ordered identifiers of line segments bounding the specified target area such that they would form a closed polygon. This differs from the 'mj_gal' routine both in the fact that it is ordered and contains only those segments comprising the border.

```
Usage:  [seg_id] = mj_gapl (a_file, area_n)
  where
      a_file is the string identifying the location of the area definition file.
            The default condition is to use the file 'MJ_AREA'.
      area_n is the numeric area identifier of the target area.
Returns:  The vector containing the qualifying line segment identifiers.  The
      lines are retrieved in the order encountered in the area definition file.
      Negative identifiers indicate reverse order of the line using the
      convention of target area on the right.
```

5.1.14 mj_gar.m

The routine 'mj_gar' retrieves the geographic extents for the user-specified area. The initialization routine 'mj_iar' must have been previously invoked to compute the ARANGE array.

```
Usage:  [lon_min  lon_max  lat_min  lat_max] = mj_gar (area_n)
  where
      area_n is the numeric area code identifier.
Returns:  The four-element vector containing the range of the area.  Note that
      the current version of the program will include any auxiliary lines that
      might have been included to form a closed polygon incorporating the coast.
```

5.1.15 mj_gld.m

The 'mj_gld' is the basic data point retrieval routine. It gets the {lon lat} data for the requested line segments.

```
Usage:  [xdeg, ydeg] = mj_gld (seg_path, seg_id, lplot)
  where
      seg_path is the string identifying the directory location of the
            'ls****.seg' line segment files.
      seg_id is the vector containing the line segment identifiers; the lines
            are retrieved in the order specified.  Negative identifiers indicate
            reverse order of line.
      lplot is a display option switch.
      0 = no plot;   1 = plot lines;   2 = also label lines
```

lhemi optional hemisphere switch for dateline wrapping. By default it will go to the Eastern Hemisphere unless this value is negative.

Returns:

xdeg is the line longitude coordinates, in decimal degrees.
ydeg is the line latitude coordinates, in decimal degrees.

5.1.16 mj_gad.m

The 'mj_gad' routine is the shortcut method used to retrieve the coordinate data that forms the closed border polygon of the requested target area. It performs the functions of getting the line segment identifiers then retrieving the data. This routine however returns the coordinates as a single line.

```
Usage:  [xdeg, ydeg] = mj_gad (a_file, area_n, lplot)
  where
      a_file is the string identifying the file location of the area definition
            file MJ_AREA.
      area_n is the vector containing the area identifiers;
      lplot is a display option switch.
            0 = no plot;    1 = plot lines;    2 = also fill lines
Returns:
      xdeg is the line longitude coordinates, in decimal degrees.
      ydeg is the line latitude coordinates, in decimal degrees.
```

5.1.17 mj_gab.m

The routine 'mj_gab' is used to retrieve the identifiers of all baseline segments bounding the specified target area identifier.

```
Usage:  [seg_id] = mj_gab (d_file, area_n)
  where
      d_file is the string identifying the location of the line descriptor file
            (MJ_LINEB).
Returns:
      seg_id - is the vector containing the line segment identifiers; the lines
            are retrieved in the order encountered in the line descriptor file.
```

5.1.18 mj_gbd.m

The routine 'mj_gbd' is used to retrieve {lon lat} data for the requested baseline segments. These are optionally plotted as points.

```
Usage:  [xdeg, ydeg] = mj_gbd (base_path, seg_id, lplot)
      base_path is the string identifying the directory location of the
            'bl****.bas' baseline node files.
      seg_id is the vector containing the baseline identifiers; the lines are
            retrieved in the order specified. Negative identifiers indicate
            reverse order of line.
      lplot is a display option switch.
            0 = no plot;    1 = plot points;    2 = also label lines
      lhemi is an optional hemisphere switch for dateline wrapping. By default
            it will go to the Eastern Hemisphere unless this value is negative.
Returns:
      xdeg is the line longitude coordinates, in decimal degrees.
      ydeg is the line latitude coordinates, in decimal degrees.
```

5.1.19 mj_gpca.m

The 'mj_gpca' routine retrieves the country code numeric identifiers of all requested points. This program uses the area range file as stored in array ARANGE. The array could have been read in from a user-generated file or computed in the initialization routine 'mj_iar'. The ARANGE array is used to simplify the search for the area in which a point is located. It can eliminate those countries it can not be in. Multiple hits still have to be tested for the defined polygons and this operation is done in the routine 'mj_ina'. Even one hit does not ensure it is uniquely in the area. In the case of multiple entries only one is returned and a message is printed for subsequent hits. Points not located in any area, e.g. land or open sea, return a NaN code.

```
Usage:  [cnty_id] = mj_gpca (xlon, ylat)
  where
      xlon  is the longitude of the point, in decimal degrees.
      ylat  is the latitude of the point, in decimal degrees.
Returns:  cnty_id  is the vector, though technically it should be only one
      element long, containing the country identifiers.
```

5.1.20 mj_ina.m

The routine 'mj_ina' determines if point is in an area polygon. Given a point and line defining a polygon, see if the point is inside the polygon (or on the border).

```
Usage:  [in_area] = mj_ina (xp, yp,  xa, ya)
      xp is the longitude of a single point, decimal degrees.
      yp is the latitude of a single point, decimal degrees.
      xa is the longitude vector of line determining a polygon; decimal degrees.
      ya is the latitude vector of line determining a polygon; decimal degrees.
            The last point need not close with the first.  Consecutive duplicate
            points are handled.
Returns:  in_area is 0 if outside polygon area; 1 if inside area or on border.
```

5.2 Utility software

The C language utility software was developed as tests to ensure the database integrity and suitability for applications development. Additional modules were developed to provide the same functionality as the Matlab routines. These have all been placed in the 'SRC' subdirectory along with associated Unix-style 'makefiles'. Global variable definitions and array allocations are done in the include file 'mj_incl.h'. This is not very sophisticated code. It was designed not to use any tricks or non-standard C language features. Working on somewhat antiquated systems has the advantage of forcing the development of software that can be easily ported. These compiled on both a DEC Ultrix system and an NT windows system using Watcom C/C++. The programs do not use any Windows or NT drivers so they are executed under just about any version of DOS.

5.2.1 mj_clr.c

'mj_clr' is a utility program which takes files of line segments (type ls****.seg) and computes the ranges, creating the line range file ('MJ_LRNG') and the line descriptor file ('MJ_LINED'). The program additionally checks the consistency of the line segment identifier in the comment and descriptor records, as well as the node count. The node count should not really be necessary in analysis but was useful in the data collection and has been carried along.

This program should be run after the initialization routine 'mj_int' and before the area range initialization routine 'mj_iar'. Note that the files are only used as part of the software and are not a required part of the basic database. This file is only an aid to creating the files two files; if the user has an alternative method for creating the files, compute away.

Usage: Command line requirements and options.

```
  mj_clr.exe  <seg_path>  [-D,MJ_LINED>]  [-N<seg_number>]  [-R<MJ_LRNG>]
    where
      <seg_path>  is the path to the directory containing the line segments.
            Default = 'SEGS/'.
      -D<MJ_LINED>            is the optional identifier for the file of line
            segment descriptor fields.  If the option is not specified the file
            'MJ_LINED' is created.
      -N<seg_number>    is a request for a specific line segment number.  Take
            care that you do not write over the existing range file unless that
            is your intent.  If no segment is specified all files are processed,
            or at least those up to the maximum number NMAX = 1000.
      -R<MJ_LRNG> is the optional identifier for the file of line segment
            ranges.  If the option is not specified the file 'MJ_LRNG' is
            created.
```

Results: Creates the files 'MJ_LINED' and 'MJ_LRNG'. See the separate sections for a more detailed description of those files.

5.2.2 mj_car.c

The routine 'mj_car' is a C language program used to extract line segments associated with a country. It then computes the rectangular longitude/latitude ranges (minimum and maximum) suitable for entry in file 'MJ_ARNG'. The program requires the numeric country code but allows the optional use of the two-character alphanumeric. If no codes are given all the countries in the 'MJ_AREA' file are processed.

As with the analogous routine 'mj_clr', this does not make use of any of the library routines in order to keep sort of an independent check of what is going on. If you want to modify it go ahead. Again, the files program and resulting files are not part of the basic database but are used with the support software.

```
Usage:  Command line requirements and options.
  mj_car.exe  [-A<area_definition_file>]  [-D<line_descriptor_file>]
            [-R<line_range_file>]  [-O<area_range_file>]
    where
      -A<area_definition_file>          is the file defining the order of
            segments constituting the maritime boundary.  The default if not
            specified is the 'MJ_LINED' file.
      -D<line_descriptor_file>          is the file containing the line
            topology.  The default if not specified is the 'MJ_LINED' file
      -R<line_range_file>     is the file containing the line segment ranges.
            The default file if not specified is 'MJ_LRNG'.
Result:
      -O<area_range_file>     is the optional specification of the output file.
            The default is 'MJ_ARNG'
```

5.2.3 mj_cbr.c

'mj_cbr' is a utility program which takes files of baseline coordinates (type bl****.bas) and computes the ranges, creating the baseline range file ('MJ_BRNG') and the line descriptor file ('MJ_LINEB'). The latter is done because we are doing all the validation here anyway.

The program additionally checks the consistency of the line segment id in the comment and descriptor records, as well as the node count. The node count should not really be necessary in analysis but was useful in the data collection and has been carried along.

```
Usage:  command line requirements and options:
  mj_cbr.exe  <seg_path>  [-N<seg_number>]  [-R<MJ_LRNG>]
    where
      <seg_path > is the path to the directory containing the line segments.
            Default = 'BASE/'.
      -B<MJ_LINEB>            is the optional identifier for the file of
            baseline segment descriptor fields.  If the option is not specified
            the file 'MJ_LINEB' is created.
      -N<seg_number>    is a request for a specific line segment number.  Take
            care that you do not write over the existing range file unless that
            is your intent.  If no segment is specified all files are processed,
            or at least up to the max number NMAX = 200.
      -R<MJ_BRNG> is the optional identifier for the file for the file of base
            point ranges.  If the option is not specified the file 'MJ_BRNG' is
            created.
```

5.2.4 mj_init.c

The program 'mj_init' performs the initialization of various paths and files for the Marine Jurisdictions suite of programs. It allows the user to set up a default environment.

```
Usage:            mj_init (computer)
  where
      computer    is a passed string argument that allows the user to select an
            environment.  The argument may in fact be any identifying string
            that the user may want to test.
```

5.2.5 mj_iac.c

Program 'mj_iac' is a C language program used to initialize areas with country identifiers. It simply reads in the file and loads up internal arrays.

Conceptually it might be a nice place for using structured elements but this version of the program is not that complicated.

```
Usage:          iopstat = mj_iac (mj_cntry_file)
  where
      mj_cntry_file  is the optional file identifier specifying
            the location of the file.
Results:    The program returns a zero value if the operation was successful.
   The global arrays CN, C2, C3 and CT are stored.
```

5.2.6 mj_ilr.c

mj_ilr' is a C language program which loads internal arrays with the values of line segment ranges stored in the file 'MJ_LRNG'. The range of each line was computed by program 'mj_clr' and stored in a file, the default system identifier being 'MJ_LRNG'. This initialization routine is called once to store the values, along with other descriptor information, in internal arrays. Technically we could run through this file when we needed it, but under the current scheme there are not that many files and we can store them to speed up things like line segment searches in ranges, with areas, line types or depths. (The latter are not yet utilized but there they are.) To further speed things up the line segment identifiers are used as indices to the array. (As a further nod to inefficiency but simplicity, we leave 0 empty and use the index directly.) NOTE: This is possible because of a couple of valid assumptions that should be stated explicitly.

1) The internal segment identifier, found on the descriptor record, is assumed correct. Program 'mj_clr' checked this on file creation but if you rolled your own and mess up you could overwrite information for a segment.
2) The segment identifiers are assumed to be pretty much contiguous, with only a few gaps, so the big array is not just air.

```
Usage:          iopstat = mj_ilr (r_file, d_file)
  where
      r_file       is the string identifying the location of the line range file
            ('MJ_LRNG'). The 'MJ_LRNG' file is created during execution of the
            program 'mj_clr.c'. The file simply contains the {lon lat} ranges
            associated with each line segment to simplify searches.
      d_file is the string identifying the location of the line descriptor file
            ('MJ_LINED'). The 'MJ_LINED' file is created during execution of
            the program 'mj_clr.c'. The file simply contains the descriptors
            associated with each line segment to simplify searches.
Results:    The program returns a zero value if the operation was successful.
      The global arrays LRANGE[4], L_TYPE, L_DEPTH and LCC[2] are stored. These
      correspond directly to the fields described in the file format. Of
      particular use is the LCC array, element [0] containing the identifier of
      the area to the left of the segment and element [1] containing the
      identifier of the area to the right.
```

5.2.7 mj_iar.c

Program 'mj_iar' is written in the C language and computes the rectangular range of an area based on the range of segments comprising the area. This is necessary initialization for the area of point (country) routine 'mj_gpca'. This is done using the segments defining the area (in 'MJ_AREA') and the pre-computed segment ranges (in 'MJ_LRNG' and loaded with 'mj_glr'). This function usually only requires one call prior to using the routine 'mj_gpca'. Alternatively the user may load the array from a previously computed or saved file.

Usage: iopstat = mj_iar (a_file)
  where
    a_file   is the string identifying the location of the area definition file 'MJ_AREA'.

Results: The program returns a zero value if the operation was successful. The global arrays 'AREA_ID' and 'ARANGE' are computed. They contain the following information.

area/country_code	lon_min, lon_max, lat_min, lat_max

5.2.8 mj_ibr.c

mj_ibr' is a C language program which loads the values of baseline coordinate ranges stored in the file 'MJ_BRNG'. The range of each line was computed by program 'mj_cbr' and stored in a file, the default system identifier being 'MJ_BRNG'. This initialization routine is called once to store the values, along with other descriptor information, in internal arrays. Technically we could run through this file when we needed it, but under the current scheme there are not that many files and we can store them to speed up things like baseline searches in ranges.

1) The internal segment identifier, found on the descriptor record, is assumed correct. Program 'mj_cbr' checked this on file creation but if you rolled your own and mess up you could overwrite information for a segment.
2) The segment identifiers are assumed to be pretty much contiguous, with only a few gaps, so the big array is not just air.

Usage: iopstat = mj_ibr (r_file, b_file)
  where
    r_file   is the string identifying the location of the baseline range file ('MJ_BRNG'). The 'MJ_BRNG' file is created during execution of the program 'mj_cbr.c'. The file simply contains the {lon lat} ranges associated with each baseline segment to simplify searches.
    b_file is the string identifying the location of the baseline descriptor file ('MJ_LINEB'). The 'MJ_LINEB' file is created during execution of the program 'mj_cbr.c'. The file simply contains the descriptors associated with each baseline segment to simplify searches.

Results: The program returns a zero value if the operation was successful. The following global arrays are created:

BRANGE[][4]	It contains the information lon_min, lon_max, lat_min, lat_max
B_CC[]	Integer area codes.

5.2.9 mj_iaa

Program 'mj_iaa' is a C language program used to initialize area attributes. The current version of the program is just a prototype used to store the something like the color. A user might use this to categorize groups of countries and use the color attribute either for selection or display.

Usage: iopstat = mj_iaa (mj_attr_file)
  where
    mj_attr_file   is the optional file identifier specifying the location of the file 'MJ_ATTR' type file.

Results: The program returns a zero value if the operation was successful. The global array 'COL_RGB' is the only array stored in the current version of the program. The program uses a table look-up to put the array in the same order and location as the area code array 'CN'.

5.2.10 mj_getcg.c

The routine 'mj_getcg' is a C language program used to extract a requested country identifier given a known property. It is the basic area-identifier conversion routine used to process user requests directly or from the other conversion specific utilities. The type of conversion is determined by the specified method.

```
Usage:  iopstat = mj_getcg (method, &code_n, code_a)
  where
      method       is the integer type of request;
            -3    given 3-character code, find numeric code;
            -2    given 2-character code, find numeric code;
             2    given numeric code, find 2 character code;
             3    given numeric code, find 3 character code;
             4    given numeric code, find text identifier.
      code_n      is the numeric country code. It may be either furnished or
            returned, depending on the method but is always passed by reference.
      code_a       is the alphanumeric country code. It may be either furnished
            or returned, depending on the method.  The user is responsible for
            supplying a string of adequate size for the method requested.
Results:    In addition to the returned code, either the numeric value or the
      string depending on the request, the program also returns the integer
      location of the element in the array, thereby enabling the user to
      retrieve the other values or attributes directly without having to do
      further look-ups.
```

5.2.10.1 mj_gcn

Program 'mj_gcn' is used to retrieve a numeric country code given the two-character ISO code. It is simply a shell that invokes the generic routine 'mj_getcg'.

```
Usage:            iopstat = mj_gcn (code_a, &code_n)
  where
      code_a       is the requested alphanumeric country code.
      code_n       is the returned numeric integer area code.
Results:    In addition to the returned value the program also returns the
      integer location of the element in the array, thereby enabling the user to
      retrieve the other values or attributes directly without having to do
      further look-ups.
```

5.2.10.2 mj_gc2

Program mj_gc2 is used to retrieve a two-character country code given the numeric code. It is simply a shell that invokes the generic routine 'mj_getcg'.

```
Usage:            iopstat = mj_gc2 (code_n, code_a)
  where
      code_n       is the requested numeric integer area code.
      code_a       is the returned two-character alphanumeric country code.
Results:    In addition to the returned value the program also returns the
      integer location of the element in the array, thereby enabling the user to
      retrieve the other values or attributes directly without having to do
      further look-ups.
```

5.2.10.3 mj_gc3

Program 'mj_gc3' is used to retrieve a three-character country code given the numeric code. It is simply a shell that invokes the generic routine 'mj_getcg'.

```
Usage:            iopstat = mj_gc3 (code_n, code_a)
  where
      code_n       is the requested numeric integer area code.
```

code_a   is the returned three-character alphanumeric country code.

Results: In addition to the returned value the program also returns the integer location of the element in the array, thereby enabling the user to retrieve the other values or attributes directly without having to do further look-ups.

5.2.10.4 mj_gct

Program 'mj_gct' is used to retrieve a country identifier text given the numeric code. It is simply a shell that invokes the generic routine 'mj_getcg'.

Usage: iopstat = mj_gct (code_n, code_a)

where

code_n   is the requested numeric integer area code.

code_a   is the returned alphanumeric text of the area/country.

Results: In addition to the returned value the program also returns the integer location of the element in the array, thereby enabling the user to retrieve the other values or attributes directly without having to do further look-ups.

5.2.11 mj_gal.c

The routine 'mj_gal' is a C language program used to extract a list of line segment identifiers associated with a specified area. NOTE: These will probably not be in the order necessary to define a closed polygon and may include segments in addition to those associated with the border.

Usage: iopstat = mj_gal (tcc, nseg, aseglist)

where

tcc   is the integer numeric target country file identifier.

nseg is an integer array allocated for the numeric count of segments returned in the list.
* on input element [0] specifies the maximum size of the provided list array.

seglist   is the integer array list of segments;
* Note that negative segment numbers indicate the segment is in reverse order for the target country on right rule. A zero entry is used as a break for countries that have separated multiple components.

Results: The program returns a zero value for successful completion. The value of the 'nseg' array will be positive, non-zero values for all components returned.

5.2.12 mj_grl.c

'mj_grl' is a C language program that retrieves the line segment identifiers of all segments whose range might overlap a specified rectangular range. Segment ranges should be initialized in the 'mj_ilr' routine.

Usage: iopstat = mj_grl (range, &nseg, seg_id)

where

range is the vector of the form [lon_min lon_max lat_min lat_max] used to specify a 'rectangular' region of interest.

nseg is, on input, the size of the 'seg_id' array being provided. On return it will contain the number of segments in the array 'seg_id'.

seg_id is the returned array that will contain the numeric line segment identifiers. Note that while the range of line may fall within the specified area, the line segment itself may not.

Results: The program returns a zero value for successful completion.

5.2.13 mj_gapl.c

'mj_gapl' is a C language program used to extract a list of ordered line segment identifiers from the area definition file. Rather than try to store the variable length lists internally, the program reads the user provided file. This also allows the user to specify alternative areas.

```
Usage:            iopstat = mj_gapl (area_def_file, tcc, nseg, aseglist)
  where
      area_def_file     is the area definition file identifier;
      tcc   is the integer identifier of the target area/country;
      nseg  is an integer array in which will be stored the count of segments in
            the list.
            [0] = number of valid segments.
            [1] = number of actual segments in a string, including breaks.
            * on input element [0] specifies the maximum of the provided list
            array.
      aseglist    is the integer array list of segments.
            * Note that negative segment numbers indicate the segment is reverse
            order for target country on right rule.  A zero entry is used as a
            break for countries that have separated multiple components.
Results:    The program returns a status of zero for successful completion.
```

5.2.14 mj_gar.c

The routine 'mj_gar' is a C language program used to extract the rectangular range of the country specified by the numeric country code. Note the values will generally be in the range 0:360+ although minor overlaps may occur (such as for countries along the Greenwich meridian) to maintain continuity). This program uses the area range file as stored in array 'ARANGE'. The array could have been read in from a user-generated file or computed in the initialization routine 'mj_iar'. The column information stored from that process is as follows:

country_code, lon_min, lon_max, lat_min, lat_max

```
Usage:            iopstat = mj_gar (code_n, range)
  where
      code_n            is the numeric area (country) code identifier.
      range is a four-element array returned with the range of the requested
            area as stored in the internal array.
Results:    The program returns a status of zero for successful completion.
```

5.1.15 mj_gld.c

'mj_gld' is a utility program which retrieves the coordinates {longitude, latitude} associated with the requested line segment identifier(s). The coordinates are nominally in decimal degrees, though there is nothing to ensure this other than how the data looks.

```
Usage:            iopstat = mj_gld (seg_path, nseg, aseglist, &npnt, xp, yp)
  where
      seg_path    is the path of the directory where line segment data is
            located.  The default file type 'ls****.seg' are set by the program.
      nseg  is the integer number of segments in 'aseglist' array being
            provided.
      aseglist is the integer array list of segments.
            * A zero entry is used as a break for countries that have separated
            multiple components.
      npnt is the returned integer number of points in segments.
            -1 indicates an array overflow condition (avoided).
            * On input this is the size of the furnished polygon arrays.
      xp    is the returned type double array of longitudes, in decimal degrees.
      yp    is the returned type double array of latitudes, in decimal degrees.
Results:    The program returns a status of zero for successful completion.
```

5.2.16 mj_gad.c

'mj_gad' is a C language program used to extract the data points from a series of line segments in a specific order so as to define a closed polygon. This version allows the user to specify their own area definition file rather that hard-wire the default.

```
Usage:              iopstat = mj_gad (aseg_fil, seg_path, tcc, &npnt, xp, yp)
  where
      aseg_fil    is the string identifying the file location of the area
            definition file, nominally 'MJ_AREA'.
      seg_path is the path of the directory where line segment data is
            located.  The default file type 'ls****.seg' are set by the program.
      tcc is the country area integer identifier.
      npnt is the integer number of points returned in polygon.
            * on input this is the size of the furnished polygon arrays.
      xp is the returned type double array of longitudes, in decimal degrees.
      yp is the returned type double array of latitudes, in decimal degrees.
Results:    The program returns a status of zero for successful completion.
```

5.1.17 mj_gpra.c

'mj_gpra' is a C language program used to extract a list of countries (their identifier actually) based on the requested point lying in the rectangular range.

```
Usage:              iopstat = mj_gpra (xy_pos, &ncr, code_n)
  Where
      xy_pos    is a two element double array containing the longitude(0) and
            latitude(1) in decimal degrees of the point being tested.
      ncr       is an integer variable into which is stored the number of
            elements returned in the list array.
            * On input this variable contains the size of the list array being
            furnished.
      code_n    is an integer array of size 'ncr' that will contain the areas
            that satisfy the general range search.
Results:    The program returns a status of zero for successful completion.
```

5.2.18 mj_gpca.c

The routine 'mj_gpca' is a C language program used to extract a list of countries (their identifier actually) based on the requested point lying in the MJ polygon.

```
Usage:              iopstat = mj_gpca (xy_pos, &ncr, code_n)
  where
      xy_pos    is a two element double array containing the longitude(0) and
            latitude(1) decimal degrees of the point being tested.
      ncr       is an integer variable into which is stored the number of
            elements returned in the list array.  Under the current
            implementation of the program this should always be the value one as
            multiple possession disputes are noted with a message.
            * On input this variable contains the size of the list array being
            furnished.
```

code_n is an integer array of size 'ncr' that will contain the areas that satisfy the country range search. The current implementation of the program should only return one value but the array must still be furnished.

Results: The program returns a status of zero for successful completion.

5.2.19 mj_gab.c

'mj_gab' is a C language program used to extract a list of baseline segment identifiers associated with a specified area. While the base points are technically just points and are not consistently grouped to form continuous lines, they have been grouped as found in the source document.

Usage: iopstat = mj_gab (tcc, &nseg, aseglist)

where

tcc is the integer identifier of the target area;

nseg is the integer count of segments returned in list;
* on input element [0] specifies the maximum size of the provided list array.

aseglist is the returned integer array list of segments;

Results: The program returns a status of zero for successful completion.

5.2.20 mj_gbd.c

'mj_gbd' is a utility program which retrieves the coordinates {longitude, latitude} of the base points associated with the requested baseline identifier(s).

Usage: iopstat = mj_gbd (seg_path, nseg, aseglist, &npnt, xp, yp)

where

seg_path is the path of the directory where baseline segment data is located. The default file type 'bl****.bas' are set by the program.

nseg is the integer number of segments in 'aseglist' array being provided.

aseglist is the integer array list of segments.

npnt is the returned integer number of points in segments.
-1 indicates an array overflow condition (avoided).
* On input this is the size of the furnished polygon arrays.

xp is the returned type double array of longitudes, in decimal degrees.

yp is the returned type double array of latitudes, in decimal degrees.

Results: The program returns a status of zero for successful completion.

5.2.21 mj_pip.c

The utility routine 'mj_pip' is a C language program used to test if a point is in a polygon. The method used here is based on an Algorithm in Glassner's "Graphics Gems", and Joseph O'Rourke's (Smith College) interpretation thereof, with a couple of modifications. Note, the original points in the polygon are destroyed. It is not necessary that the polygon is closed but if it is not the end points will be connected.

Usage: iopstat = mj_pip (xy_pos, nvrt, Px, Py)

where

xy_pos is a two element double array containing the longitude(0) and latitude(1) in decimal degrees of the point being tested.

nvrt is an integer value specifying the number of points in the arrays being passed for the polygon coordinates.

Px is the type double array of longitude coordinates defining the polygon to be tested.

Py   is the type double array of latitude coordinates defining the polygon to be tested.

Results: The integer status is returned a value of one (1) if the point is in the polygon, zero otherwise.

5.3 Test programs

The test programs are written in the C language and exercise the components of the utility software. The also illustrate the manner in which some of the initialization routine are used and how the parameter list are passed to the various functions.

The purpose of each test program is summarized in Appendix O. A brief explanation and example is also provided if the program is invoked without any arguments. For example,

```
    mjt_01.exe  or   mjt_02.exe
```

The tests are neither exhaustive nor foolproof, merely some examples.

5.4 Application prototypes, conversion programs

As an extension of the test programs some routines were developed which allow the porting of the MJDB data to other analysis and/or display systems. These are written in the C language and make use of the software utilities described in an earlier section. The programs are described briefly in the sections that follow.

5.4.1 mja_gxy.c

This program is an extension of the border retrieval by area code test. Here the coordinates are retrieved then output in an ASCII list format compatible with the Generic Mapping Tool (gmt) software package. In the normal mode of operation the file identifier is formed using the two-character area identifier as the root filename. The optional second argument allows the user to put everything in a single multi-segment file with identifier 'MJ_GMT.gxy' segment. The segment comprising each area will be separated by the gmt default multi-segment separator, the '>' record.

Usage: mja_gxy.exe area_code_n [multi]

Results: The files contain one record per {longitude latitude} coordinate pair. The single area files might also be used with any software, such as Matlab or Surfer, that accepts list input.

5.4.2 mja_msg.c

The 'mja_msg' program is used to extract the border polygon for an area and create {count latitude longitude} coordinate records in message format files. The file created is suitable for use with various software packages but was developed specifically for those products such as 'msg2vec' which use the message format. This version of the program only processes one area per message file. It also differs from the GMT format version in that the user can specify the segment file path.

Usage: mja_msg.exe seg_file_path area_code_n

Results: The program creates one file, using the two-character area code as the root file identifier, for each area requested.

5.4.3 mja_vpf.c

The 'mja_vpf' program extracts border coordinates for an area and creates Vector Product Format (VPF) LINE files. Like the message format converter the user can specify the segment file path. The two-character area code is used to form the root file identifier and only one file is created per area request. The resulting file can be read directly into the DMAMUSE product distributed by NIMA and used at NAVOCEANO. There is a lot of header information in the file. As this is output before the coordinate data it is imperative that the node count and area ranges, as stored in the 'MJ_LINED' and 'MJ_ARNG' files, be accurate.

Usage: mja_vpf.exe seg_file_path area_code_n
Results: The current version of the program generates the LINE format. Extensions could be made to generate some of the other forms of the data. A complete description of the Vector Product Format is contained in the Department of Defense 'Interface Standard for Vector Product Format'; the document is available from a number of sources on the World Wide Web.

5.4.4 mja_lshp.c

Program 'mja_lshp' is used to extract line segments and create files in the ESRI 'shapefile' format. A 'shapefile' has become something of a de facto standard for transporting geographic information in and out of various software packages. The data can be imported directly into a package such as ESRI's 'ARCVIEW' as well as several other GIS packages. The files, and there are three of them to keep track of, are not visually readable ASCII but rather are icky machine independent binary.
As mentioned, there are three types of files associated with the 'shapefile'. The file extensions are considered inviolate.

a) master data file, .shp
b) index file, .shx
c) dBASE file, .dbf

And now a list of caveats:
This is an example of how to create 'shapefiles' from the data. It makes no claims for being completely general purpose or even the best way of doing it. It is meant to be a simple and understandable illustration of how it can be done.

There are a couple of ways of going about this. The main problem is a 'shapefile' requires a prior knowledge of how many records and bytes are going to be in it. (Either that or direct access/keyed writes, which are not machine independent.) So we assemble a list the first pass. Again, this could use memory allocation and all that wonderful C language stuff but there are only a limited number of segments anyway. Finally, we could use the 'mj_gld' function and get the actual number of points. Or if one feels that would take too long we could get the information from the 'MJ_LINED' record, but that also would require some search. I've opted for the slow method as it allows me to do some range checking and make sure it looks like I will eventually see it in ArcView.

This is written primarily for the PC based machine. Or more correctly the endian conversion routines are currently set up for running from a PC. This implementation is used to pass the line segments into the file. The attributes that go into the database portion are completely arbitrary but I've chosen to at least carry along the topological information concerning the right- and left-hand area codes.

Usage: mja_lshp.exe seg_file_path line_code_n
Results: Because almost any combination of line segments could be requested the file identifiers are hard-wired to the following:

MJ_LS.dbf
MJ_LS.shx
MJ_LS.shp

5.4.5 mja_pshp.c

Another conversion program, 'mja_pshp', is used to extract a border for an area and create ESRI 'shapefiles' for the polygon and 'midpoint. Whereas the program 'mja_lshp' extracted line segments, this application extracts the border polygon. The two-character area/country identifier is used to form the root of the file identifier; it is appended with the character 'p' for the point 'shapefile' and 'a' for the polygon 'shapefile'. The location of the point in the point-type 'shapefile' is the computed midpoint of the area. In most cases this should lie within the outlined area but there is no guarantee.

There are three types of files associated with the 'shapefile'. The file extensions are considered inviolate.

a) master data file, .shp
b) index file, .shx
c) dBASE file, .dbf

If one does one country per file then name can be used. However, the definition allows multiple polygon 'shapes' in a file. So what do we call this. It may be a mute point as the 'shapefile' is not topological and will duplicate any common lines. I am therefore going with one country per area, or one file per area if one chooses to think of it in that manner. It should be easy enough (famous last words) to extend it if multiple areas must be put in the same file. It could get huge. The list of caveats in the earlier line-style 'shapefile' discussion applies here as well.

```
Usage:          mja_pshp.exe   seg_file_path   area_code_n
```

Results: The program produces six files using the two-character area/country code as the root of the file identifier. They are of course binary.

6.0 Documentation

This document is the primary source of documentation for this phase of the project. It has been placed in the 'DOC' subdirectory in both Microsoft Word 97 and ASCII text formats. Supporting figures showing the boundaries were not included in the MJDB documentation for portability reasons, but have been placed in the DOC subdirectory. The files were generated with the GMT program and are stored in the PostScript language. The support software has descriptions and comments embedded in the code.

7.0 Discussion and comments

This section is a catchall for everything that did not logically fit in some other section of the documentation. It could just as easily been put in the form of questions and answers except some of the issues do not really have answers.

<u>Reliability of the line segments</u>

I hope the disclaimer made it clear this is not the last word on the Marine Jurisdictional boundaries. First, the defining of boundaries is an on-going process, both between established nations and arising as new nations are formed. Second, even if neighboring parties agree on a boundary, the rest of the world may not accept it. Finally, there seems to be a lack of written material for many of the boundaries.

Most of the line segments were used in preparation of the Ross/Fenwick Marine Jurisdictions Chart and represented the boundaries as best determined at that time. It was more than adequate for the scale of the chart on which it was displayed. In using and updating that data set, and looking at the finer points of the topological relationships, it became clear that there were many areas where it was difficult to determine where the boundaries really were located. This release represents the best effort to resolve the inconsistencies with the data available.

<u>Line segment data accuracy</u>

If this were a question it would be a trick question. Strictly speaking, there are three properties of the data: accuracy, precision and resolution. The subject of accuracy was actually touched on in the previous section under reliability. How well does the data represent reality? For the majority of the lines the location is not only not well documented but the digitization process has been shown to have missed known points by anywhere from 100 to 1000 meters. The coordinates of points along a boundary have been stored to 0.000001 degrees. At the equator this corresponds to about 0.1 meters, which should be enough precision for most applications. The resolution of most current treaties seems to be about 0.1 minutes (~200 meters). The result of all this is the database can adequately record the boundaries when they are available.

<u>Line segment data volume</u>

The line segments probably contain way too much data. As stated, they were probably digitized from some source chart and the coordinates were recorded about every 1000 meters. Where boundaries have been defined as a straight line connecting two widely separated points this appears to be a clear case of over kill. The condition is also apparent for the 200 nm. limit of islands in the open ocean; there are a lot of point in the line segment. Yet one should not be too hasty in decimating the coordinates defining a line segment. First, one has to be careful not to throw away a corner or a defined point. Perhaps more importantly, resulting long, unconstrained, 'straight' boundary lines can assume some decidedly inaccurate shapes and positions on different map projections. After taking all this into account, the data has been retained as recorded for the originally digitized data set. Where the boundary has been defined by a set of discreet points only those points have been recorded and no intermediary points are included.

Some of the long, open ocean arcs, such as most often found in the Pacific, have been trimmed to provide a better junction with a treaty-defined line. At some point the arcs need to be recomputed. The island 'basepoints', either actual or implied by the arc, are in most cases unknown.

<u>Boundary definitions</u>

Almost all the explicit boundary data has come from the DoD document. While this seems to be the best current source by far it needs to be used with some care. There are more than a few errors, inconsistencies and unclear passages. In some cases it is a matter of stating different values in different sections. This may be a case of disagreement by two parties but in the case of treaty agreements seems more likely a misprint. There are errors of sign and hemisphere. In most cases the errors, when not immediately obvious, become apparent when viewed on a chart.

As stated earlier, even with the DoD document, considerably less than half the countries have boundaries attributable to a source document. It seems like somebody must be keeping track of all this. The United Nations would seem like a logical candidate. Contact was made with the office and persons given below.

Division for Ocean Affairs and the Law of the Sea
Office of Legal Affairs
United nations
Two United Nations Plaza
Room 0472
New York, NY 10017
Mr. Vladimir Jares
jares@un.org
212/963-3943
Mr. Antonio Escudero
212/963-3948

They have plans, wonderful plans. The will not only store all the boundary data, baseline data and treaties but also keep it current and perhaps even accessible. But it seems to be still just a plan.

Conflicting border claims

As stated earlier, the process of achieving universal, or at least global, acceptance of all boundaries seems likely to be fulfilled. In the large middle ground are the boundaries that may not be official but are at least observed. There are a few boundaries, the parties to which are openly contentious to a lesser or greater degree. The data base allows the border segments to be connected in any fashion so if you do not think they boundaries are correct, or they change, simply redefine the border. Some of the areas that have unsettled or conflicting claims are listed below. These are not meant to be recognition of claims, merely an indication that a conflict does exist.

Argentina / Great Britain (Falklands)
Bahamas / United States of America
British Ocean Territories / Maldives
China / Taiwan and other neighbors
Haiti / United States of America (Navassa)
Japan / Russia
Namibia / South Africa
North Korea / neighbors
Philippines / neighbors
Vanuatu / neighbors
Vietnam / neighbors

Missing boundaries and territorial waters

For all practical purposes the boundaries for areas such as the Mediterranean Sea, Baltic Sea, Red Sea, Black Sea and Persian Gulf should be considered as non-existent. They have been carried along where available but they are far from complete. In many cases these claims fall under the more appropriate classification of territorial seas and while they might be considered a subset of the more general Exclusive Economic Zone, they have not been specifically addressed in this project.

Another particularly difficult area is the South China Sea and any border coming anywhere near China. This includes the interpretation of the Philippine boundary. This is primarily an issue of lack of data. In the current version of the data set the boundaries again fall into the category of 'best interpretation based on existing evidence'.

NIMA port data

As discussed in another section a subset of the NIMA World Port Index has been provided with the data. Errors have been discovered in the current NIMA release. Some of the more notable errors, such as the wrong hemisphere, have been corrected. There may be other errors so use the product with care.

Polygons and closure

In cases like isolated islands it is pretty straightforward to define a closed polygon for the area of the maritime claims. When displayed and then overlain with the land everything looks great. Even for large areas such as the United States of America and Russia it is possible to construct the lines segments comprising the border into a polygon such that the superimposition of the landmass masks the large 'gaps' in the maritime border. Sadly, there are a few countries where these assumptions either do not work or look really bad when displayed. The countries of Argentina and Angola come to mind; there are others. To generate at least a nice looking map some of the territorial boundaries on land were incorporated into the polygon definition. They follow the same rules as all the other line segments. These special segments are numbered with the identifiers beginning at 951. They may be overkill for some users needs, not enough for others. The assumption regarding the closure by connecting of end points is not merely cosmetic however. It is also implicit in the point in polygon searches. A bad approximation to the coast can result in a 'hole' between the boundary and the coast of the country. This is not good. It should be equally obvious, but will be stated anyway: with the current database the point in polygon search will be inaccurate for points on land.

Updating the database

In view of the fact there is not yet a central facility to maintain the database users may want to do their own updates, interpretations or modifications. The follow paragraphs contain some methodology for performing various types of updates.

Replacing a line with a new line: Suppose one wishes to decimate the number of coordinates in a line. Or perhaps needs to add some coordinates to achieve a better representation. An uncertain line might also be replaced with a new line define by treaty. At any rate, assuming the line is bounded only by the same two areas/countries check to ensure the topology of the area codes. If the line is going in the opposite direction the order of the codes will be reversed. Check to make sure the end points of the new line segment match those in the segment being replaced. Or more correctly make sure the endpoints correspond to the junction with segments at either end of the new segment. (The junction may have been an indeterminate point and now need updating in several files.) Update the line segment file's descriptor record field containing the number of points. Be sure to note in the appropriate fields the source of the information and any comments. The file is replacing an existing file of the same name, as determined by the line identifier, so not much more has to be done there. If the direction of the line has been reversed, the area border definition file ('MJ_AREA') must be modified. There will be two countries affected and the entries for the line segment identifiers must have the signs reversed. The positive will become a negative and vice versa. Two other less critical files might also need updating. The line definition file ('MJ_LINED') should be updated so the line being updated has record information matching that on the file descriptor record. The line range file ('MJ_LRNG') should also be updated to reflect the new longitude and latitude range of the file. These files may be either edited or recreated using the utility program 'mj_clr.c'. Finally the area ranges for the two affected countries may have changed. Again this file is not critical, especially if the changes have been small, and the file 'MJ_ARNG' may be either edited or recreated using the utility program 'mj_car'.

Creating a new line segment: There are a couple of reasons a new line segment may be needed. New data may become available defining a border not previously included in the database. This would include the formation of a new country. Or new coordinates might become known along part of existing boundary and splitting the current segment into two parts would best incorporate these changes. The first step is to determine a new identification number for the line segment. The best procedure is to fill in the holes but most obviously choose one not already in use. Form the file identifier and create the file in the 'SEGS' subdirectory. The format of the file is described in an appendix but it is just as easy to use an existing file as a template. The next steps are similar to those described in the preceding discussion so are just summarized here.

1) Ensure the internal line segment identifiers match that used for the file.
2) Determine direction of the line and appropriate area identifiers from the area code definition file ('MJ_CNTRY').
3) Enter left and right area codes; the fields on the descriptor record are mandatory.
4) Specify the number of coordinate points in the file.
5) Fill in the source and other fields on the descriptor record as appropriate.
6) Include the points in the file, longitude and latitude decimal degrees.
7) Update the area border definition file ('MJ_AREA'). Locate the identifiers of the two segments that the new segment will connect and insert the new identifier in the stream. Make sure the sign is correct for keeping the target country on the right. Remember, usually two areas will need to be updated and there is only one definition record per area. If the border is with the open ocean or a country not in the database then only one area may need updates. If this is an entirely new country make sure to include any additional fields of information as described in an appendix or use one of the other areas as a template. The is not other way of maintaining the area border definition file.
8) Update the line segment range file 'MJ_LRNG'. This is more important when a new segment is being defined. The file may be either edited or recreated with the program 'mj_clr.c'.
9) Update the line segment description file 'MJ_LINED'. This is more important when a new segment is being defined. The file may be either edited or recreated with the program 'mj_clr.c'.
10) Update the area range file 'MJ_ARNG'. The file may be either edited or recreated with the program 'mj_car.c'.

If an existing segment is being split into two, or more, parts the above process may need to be repeated several times. Make sure the coordinates at the end of the new segment agree with those at the junction with adjacent segments within the tolerance desired.

Software compatibility

The various software modules were developed using some pretty old hardware, operating systems and vendor software. It should be upward compatible without too many changes being required. That is not to say that some changes might not make things a lot more efficient. For instance, there are many new features in the Matlab 5.0 release that either replaces obsolete functions used here or greatly simplify things. Similarly for the functionality employed in the C language programs. The guiding principle was, if I had old tools some other user would probably have something even older. Until the applications and operational requirements become more specific everything was kept as simple and portable as possible.

Future developments

To go beyond the statement in the disclaimer, the initial version of the product is probably the best available at the time of its release. It is quite probably the only version. As it gets out in the public domain there will undoubtedly be modifications, not only to the access, analysis and display software, which is to be wholeheartedly encouraged, but also to the form and content of the boundary data itself. This raises issues of long term maintenance and a central source or repository of information. On the distant horizon the United Nations appears to be willing to assume this role. It was not clear in my conversations with personnel there if the current charge included unrestricted public access to the information. Major GIS vendors may also have financial interests in distributing a maritime boundaries product for customers in a number of fields. The intention is that this product can fill the immediate need for a database of maritime boundaries and serve as a starting point for future developments in the area. But some thought and planning should be given to the direction of that development starting now.

Acknowledgments

The impetus for this project was provided by RADM Richard Pittenger, USN (Ret.), Associate Director for Marine Operations at the Woods Hole Oceanographic Institution (WHOI), who gathered the various interested parties together and provided the necessary WHOI support. Ms. Emma Dieter, Program Manager for Ship Operations at the National Science Foundation helped provide funding through the Office of Naval Research contract #N00014-98-1-0276. CDR James Trees, Operations Officer at the Naval Oceanographic Office (NAVOCEANO), provided not only manpower and material support, but also the operational need for the product. QM Glenn Paul spent a number of days providing some insight into the operations at NAVOCEANO and outlining the role a marine jurisdictions database should play. Additional thanks go to Dr. David Ross/WHOI and Ms. Judith Fenwick/WHOI, the authors of the original Maritime Claims chart on which this project was based, for their continuing interest and assistance. Acknowledgement is also given to the National Imagery and Mapping Agency, a subset of whose World Port Index is included with the database for added functionality.

References

DoD 2005.1-M, 'Maritime Claims Reference Manual' (electronic version), Department of Defense, Washington, D.C., Jan 1997.

DoD MIL-STD-2407, 'Interface Standard for Vector Product Format', Department of Defense, Washington D.C., 27 April 1996.

ESRI White Paper, 'ESRI Shapefile Technical Description', Environmental Systems Research Institute, Redlands, CA, May 1997.

Glassner, Andrew S., 'Graphics Gems', Academic Press, San Diego, CA, 1990.

Fenwick, Judith, 'International Profiles on Marine Scientific Research', WHOI Sea Grant Program, Woods Hole Oceanographic Institution, Woods hole, MA, 1992.

National Imagery and Mapping Agency, 'World Port Index', Marine Navigation Department, Washington, D.C., 1998.

Ross, Dr. David A. and J. Fenwick, 'Maritime Claims and Marine Scientific Research Jurisdiction', WHOI Sea Grant Program, Woods Hole Oceanographic Institution, Woods Hole, MA, 1992.

Appendix A: Line segment file format

Record type #1: Optional comment record.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated. The '#' first character is all that is required to identify the record as a comment. The rest can be anything the user wants; what is specified here is just the convention adopted for the first stage of this project.

001:001	Hash character '#' indicating optional comment record.
002:007	Integer line identifier, leading zero-filled.
010:012	Integer data source identifier. [See Table I: Data source.]
015:end	Optional comments. The contents of this field may be almost anything. However, in this implementation I have chosen to start the field with mnemonic 'll|rr', where 'll' is the two character code of the country to the left moving along the line. 'rr' is the code of the country to the right.

Record type #2: Line descriptor

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:001	Character 'D' indicating descriptor record.
002:007	Integer line identifier, leading zero-filled.
009:013	Integer line-type identifier. [See Table II: Line type codes.]
016:020	Integer depth of line feature. This field really is not used in the current implementation so it is usually 0. It was defined nominally as depth to allow flexibility for some attribute assignment at a later time.
023:026	Integer left side area identifier.
028:031	Integer right side area identifier.
033:036	Integer number of coordinates in line feature.
040:040	Integer display code; 0=do not display line. This is another field not completely utilized. The intent is that some segments, such as closing land boundaries, are part of the border but should not be displayed as such. A value of 1 or some other non-zero value could be used as an additional line attribute.

Record type #3: Line feature coordinates blank separated {longitude latitude} coordinate pairs, one pair per record.

Example:
```
#000001  001  AU|NC
D000001 00061  00000  0036 0540 0407
 158.531799 -25.065601
 :
 :
```

Table I: Data sources.

001	Data from LOTS tape
002	Missing segment constructed from 001
003	USA/Russia Maritime Agreement 01 Jun 1990 'LOS Bulletin #17 pp. 15-21'
004	WDBII via gmt extraction
005	Fr. Rep. (New Caledonia)/Solomon Islands 'LOS Bull. 18 & 19 w/corr'

006 DoD 2005.1-M, "Maritime Claims Reference Manual (electronic version), Department of Defense, Washington, D.C., Jan 1997.
007 "Times Atlas of the World, Comprehensive Edition", New York Times Book Co., New York, N. Y., 1980.
008 "Columbia Lippincott Gazetteer of the World", Columbia University Press, Morningside Heights, N. Y. 1961.
009 Electronic Mail, CDR. James Trees, NAVOCEANO, 15 May 1998.

Table II: Line type codes

The line type codes are an extension of those supplied with the digitized data set of source [1]. There is some indication that these originally came from the World Data Bank II (WDB-II) ranking guide, which apparently had a Law of the Sea Data Bank as a subset. The actual source is unknown however, as is the method of categorization of the original data set. The following rank codes were either used or added for this project.

Law of the Sea Data Bank	
01	International boundary (added for this project)
08	Baseline points (added for this project)
31	200 nautical mile limit
41	Agreed upon territorial sea boundary
42	Agreed upon continental shelf boundary
43	Agreed upon maritime boundary
61	Hypothetical lines to 200 nautical miles
81	Joint development zone
84	Joint fishing zone
85	Joint protected zone
91	Exclusive Economic Zone
92	Straight lines
93	New US-Canada line
94	Fishery conservation zone

Appendix B: MJ_CNTRY file format

Record type #1   Area code definitions.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:005	Right justified or leading zero-filled integer area code.
007:008	ISO-3166 two-character area code.
010:012	Three-character area code.
014:083	Full country identification. Actually, this can be whatever you like. In most cases this seems to be the official name. It could be changed to the popular name or to use lower case. There is no specific limit to how much you put on the record but the support software currently allows a maximum of 69 characters.

Example:

```
   24 AO AGO ANGOLA
   36 AU AUS AUSTRALIA
 1518 M4 M04 MACQUARIE ISLAND (AUSTRALIA)
 :
 :
```

Appendix C: MJ_LRNG file format

Record type #1   Line ranges.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:004	Right justified or zero-filled integer line segment identifier.
006:016	Minimum longitude of points in the line segment.
018:028	Maximum longitude of points in the line segment.
030:039	Minimum latitude of points in the line segment.
040:049	Maximum latitude of points in the line segment.

Example:

```
0002  160.442322  164.880402 -55.247501 -51.114498
0003  168.779007  171.120697 -32.300598 -30.948101
0004  287.187202  289.456093  21.897001  25.038301
  :
  :
```

Appendix D: MJ_LINED file format

Record type #1: Line descriptor

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:001	Character 'D' indicating descriptor record.
002:007	Integer line identifier, leading zero-filled.
009:013	Integer line-type identifier. [See Table II: Line type codes.]
016:020	Integer depth of line feature. This field really is not used in the current implementation so it is usually 0. It was defined nominally as depth to allow flexibility for some attribute assignment at a later time.
023:026	Integer left side area identifier. This is the area (country) code as defined in the file MJ_CNTRY.
028:031	Integer right side area identifier. This is the area (country) code as defined in the file MJ_CNTRY.
033:036	Integer number of coordinates in line feature.
040:040	Integer display code; 0=do not display line. This is another field not completely utilized. The intent is that some segments, such as closing land boundaries, are part of the border but should not be displayed as such. A value of 1 or some other non-zero value could be used as an additional line attribute.

Example:

```
D000002 00061  00000  0554 1518 0277   1
D000003 00061  00000  0554 0574 0095   1
D000004 00061  00000  0796 0044 0108   1
  :
  :
```

Appendix E: MJ_AREA file format

Record type #1: Optional comment record. The record is primarily an aid to expedite the identification of an area. They may appear anywhere in the file but obviously are most useful if tied to and located somewhere in the vicinity of the boundary definition record.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated. The '#' first character is all that is required to identify the record as a comment. The rest can be anything the user wants; what is specified here is just the convention adopted for the first stage of this project.

001:001 Hash character '#' indicating optional comment record.
002:007 Integer line identifier, leading zero-filled.
010:011 Two-character area (country) code. The codes are defined by the ISO-3166 standard and are contained in the file 'MJ_CNTRY'.

Record type #2: jurisdictional limit boundary definition.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated. This record is as long as necessary and must not be broken into multiple records.

001:001 Character 'A' indicating area definition record.
002:007 Integer line identifier, leading zero-filled.
009:013 Integer area type identifier. [See Table II: Line type codes.] This field is meant to allow some type of description for the type of area being defined. As such it is probably a function of the type of line segments used to construct the area. That is the long-term plan anyway; it has not been rigorously implemented in this version of the project.
016:020 Integer attribute of line feature. This field really is not used in the current implementation so it is usually 0. It was defined nominally as depth or time to allow flexibility for some attribute assignment at a later time.
022:026 Integer line-segment identifier of the first segment in the border. The convention defines the associated area to the right as one moves from the first to subsequent points of the line segment. A negative value is used to indicate that one must travel 'backward' along the line.
028:032 Integer line-segment identifier of the next section of the border. The sense of the line is continually defined so the associated area is on the right.
:
:

List as many segment identifiers as needed to describe the border.

Example:
```
#001509  A6
A001509 00061  00000  0692  0687  0688
#001517  A7
A001517 00061  00000 -0385 -0313  0312  0311  0757  0758
#001526  A8
A001526 00061  00000   0741
```

Appendix F: MJ_ARNG file format

Record type #1   Area ranges.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:004	Right justified or zero-filled integer area identifier.
006:016	Minimum longitude of points in the area border.
018:028	Maximum longitude of points in the area border.
030:039	Minimum latitude of points in the area border.
040:049	Maximum latitude of points in the area border.

Example:

```
1509  324.466095  339.133900  33.547802  43.032204
1517   88.788101   95.696205   4.021400  15.767200
1526  342.266899  348.998000 -11.284200  -4.618300
 :
 :
```

Appendix G: Base point file format

Record type #1: Optional comment record.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated. The '#' first character is all that is required to identify the record as a comment. The rest can be anything the user wants; what is specified here is just the convention adopted for the first stage of this project.

001:001 Hash character '#' indicating optional comment record.
002:007 Integer line identifier, leading zero-filled.
010:012 Integer data source identifier. [See Table I: Data source.]
015:end Optional comments. The contents of this field may be almost anything. However, in this implementation I have chosen to start the field with the area topology as define for the line segment. As these are internal to the country the area identifier should be the same for both fields.

Record type #2: Baseline descriptor

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:001 Character 'B' indicating descriptor record.
002:007 Integer line identifier, leading zero-filled.
009:013 Integer line-type identifier. [See Table II: Line type codes.] This field could be better used to define the type of baseline. As it is, the data entered thus far does not mean much.
016:020 Integer depth of line feature. This field really is not used in the current implementation so it is usually 0. It was defined nominally as depth to allow flexibility for some attribute assignment at a later time.
023:026 Integer left side area identifier. This field should always contain the area code for the selected country.
028:031 Integer right side area identifier. This code should match the previous field.
033:036 Integer number of coordinates in baseline feature.
040:040 Integer display code; 0=do not display line. This is another field not completely utilized. This field has been set to a value of zero for the base points.

Record type #3: Base point coordinates blank separated {longitude latitude} coordinate pairs, one pair per record.

Example:
```
#000055  006  FJ|FJ Archipelagic baselines
B000055 00008  00000  0242 0242 0034   0
 180.856667 -16.091667    1
 181.068333 -16.746667    2
 :
 :
```

Appendix H: MJ_BRNG file format

Record type #1   Base point ranges.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:004	Right justified or zero-filled integer baseline identifier.
006:016	Minimum longitude of base points in the file.
018:028	Maximum longitude of base points in the file.
030:039	Minimum latitude of base points in the file.
040:049	Maximum latitude of base points in the file.

Example:

```
0001  357.832778  368.641667  35.078889  37.096667
0002   11.672222   13.370833 -16.544167  -8.583333
0003  301.749722  304.133333 -35.633333 -34.200000
 :
 :
```

Appendix I: MJ_LINEB file format

Record type #1: Baseline descriptor

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:001	Character 'B' indicating descriptor record.
002:007	Integer baseline identifier, leading zero-filled.
009:013	Integer line-type identifier. In the current implementation this value has been set to a generic baseline indicator.
016:020	Integer attribute of baseline feature. This field really is not used in the current implementation so it is usually 0. It was defined nominally as depth to allow flexibility for some attribute assignment at a later time.
023:026	Integer left side area identifier. This is the area (country) code as defined in the file MJ_CNTRY.
028:031	Integer right side area identifier. This is the area (country) code as defined in the file MJ_CNTRY.
033:036	Integer number of coordinates in line feature.
040:040	Integer display code; 0=do not display line. This is another field not completely utilized. The intent is that some segments, such as closing land boundaries, are part of the border but should not be displayed as such. A value of 1 or some other non-zero value could be used as an additional line attribute.

Example:
```
B000001 00008  00000  0012 0012 0076   0
B000002 00008  00000  0024 0024 0008   0
B000003 00008  00000  0032 0032 0015   0
 :
 :
```

Appendix J: MJ_ATTR file format

Record type #1: Area attributes

Note: The format, while nominally column aligned, should be treated as blank separated fields.

001:005 Integer numeric country/area code.

007:008 Two character ISO country codes. These should not meant to be used from this file; they are included to facilitate reference.

010:010 The opening bracket indicates the first (and at this point only) attribute, the color component associated with the area. The normalized RGB triplet follows this field indicator. The brackets {'[]') are used to enclose the color attribute values.

Example:

```
 1010 A0 [0.00 1.00 1.00]
 1011 A1 [0.00 1.00 1.00]
 1012 A2 [0.00 1.00 1.00]
 1013 A3 [0.00 1.00 1.00]
   20 AD [1.00 1.00 1.00]
  784 AE [1.00 1.00 1.00]
    :
    :
```

Appendix K: MJ_LOC file format

Record type #1   Point of interest location records.

Note: The column delineation is primarily for editing and verification. All processing operations are expected to operate on fields which are blank separated.

001:011	Longitude of point, in decimal degrees.
013:022	Latitude of point, in decimal degrees.
024:027	Right justified or leading zero-filled integer area code.
029:030	ISO-3166 two-character area code.
032:083	Full point identification. Actually, this can be whatever you like. In most cases this seems to be the official name. It could be changed to the popular name or to use lower case.

Example:

```
  1.500000  42.500000 0020 AD ANDORRA
 53.000000  25.533333 0784 AE ABU AL BU KHOOSH
 54.383333  24.533333 0784 AE ABU DHABI
        :
        :
```

Appendix L: Summary of Matlab routines

```
Contents
% mj_gab    get area (country) baseline identifiers.
% mj_gac    get area (country) code.
% mj_gad    get area (country) data.
% mj_gal    get area (country) lines.
% mj_gapl   get area border (segment_identifiers).
% mj_gar    get area (country) range.
% mj_gbd    get baseline data.
% mj_gcn    get area (country) numeric code identifier.
% mj_gc2    get area (country) two character identifier.
% mj_gc3    get area (country) three character identifier.
% mj_gct    Get area (country) text.
% mj_gld    get line data.
% mj_gpca   get (country) area of point.
% mj_grl    get range lines.
% mj_iaa    initialize area (country) attributes.
% mj_iac    initialize area (country) codes.
% mj_iar    get area (country) ranges.
% mj_ilr    initialize line ranges.
% mj_ina    determine if point is in area polygon (utility).
% mj_init   initialize default file locations, etc.
```

Appendix M: Example using some of the Matlab functions stored in the MFILES subdirectory

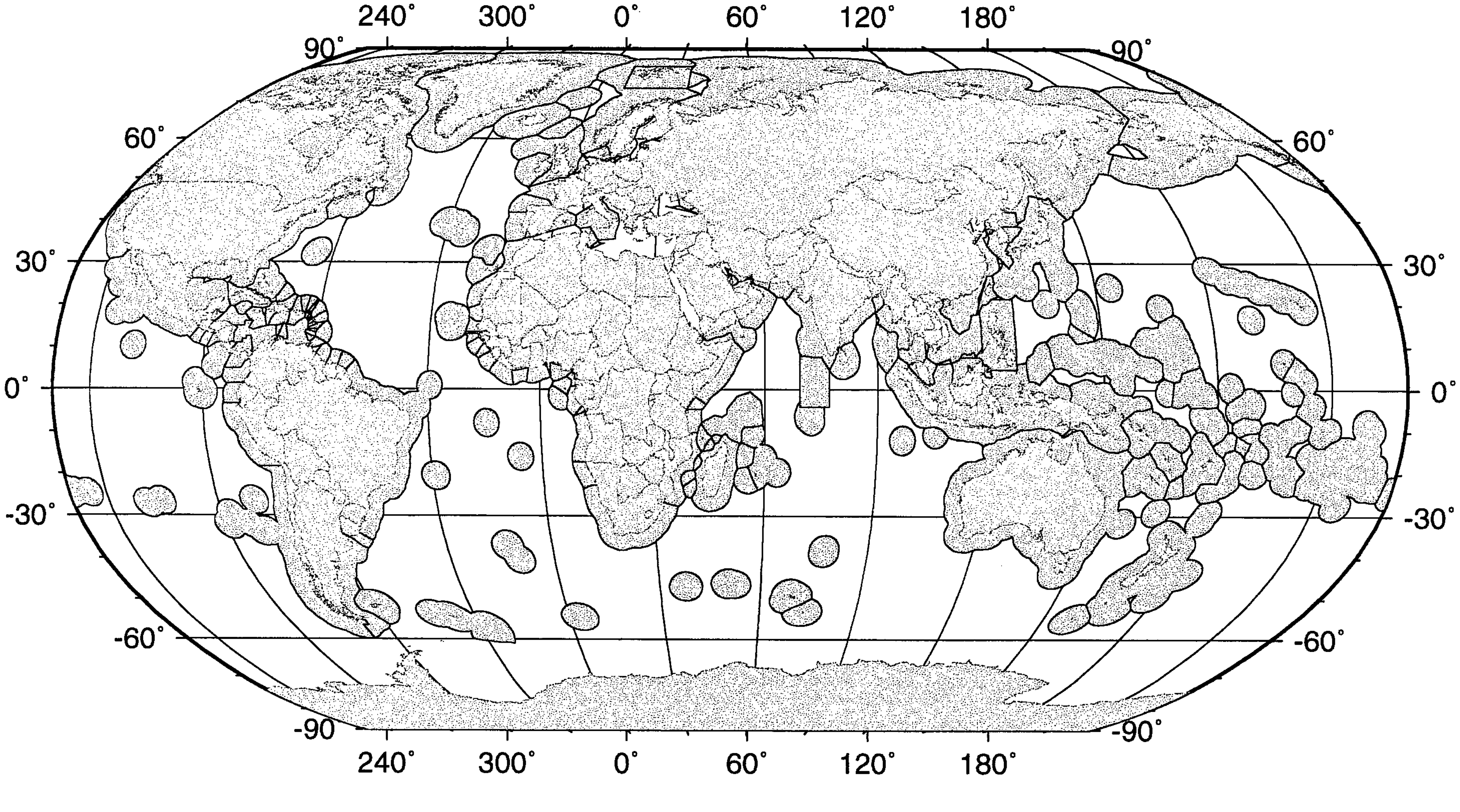
```
     % This example assumes a Unix environment and
     % operating in the parent MJDB directory.

     % Add the MFILE location to the path.
PATH = path;
path ('./MFILES', PATH);

     % Initialize with the default environment.
mj_init
mj_iac
mj_ilr
mj_iar
pause

     % Get the area identifier of Norway.
kid = mj_gcn ('NO')
     % Verify to make sure this is the right area.
mj_gct (kid)
pause

     % Get all line associated with the country.
kseg = mj_gal ('MJ_LINED', kid)
pause

     % Get the line segments and plot them.
[xd, yd] = mj_gld ('SEGS/', kseg, 2);
pause

     % Get the ordered border and plot it.
[xd, yd] = mj_gad ('MJ_AREA', kid, 2);
pause

     % Get the base points and add them to the plot.
kseg2 = mj_gab ('MJ_LINEB', kid);
[xd2, yd2] = mj_gbd ('BASE/', kseg2, -2);
pause

     % Get range of Norway.
range = mj_gar (kid)
pause

     % Now expand range slightly and get all lines
     % in the area.
kseg3 = mj_grl ([range(1)-5  range(2)+5  range(3)-5  range(4)+5])
[xd3, yd3] = mj_gld ('SEGS/', kseg3, -1);
pause
```

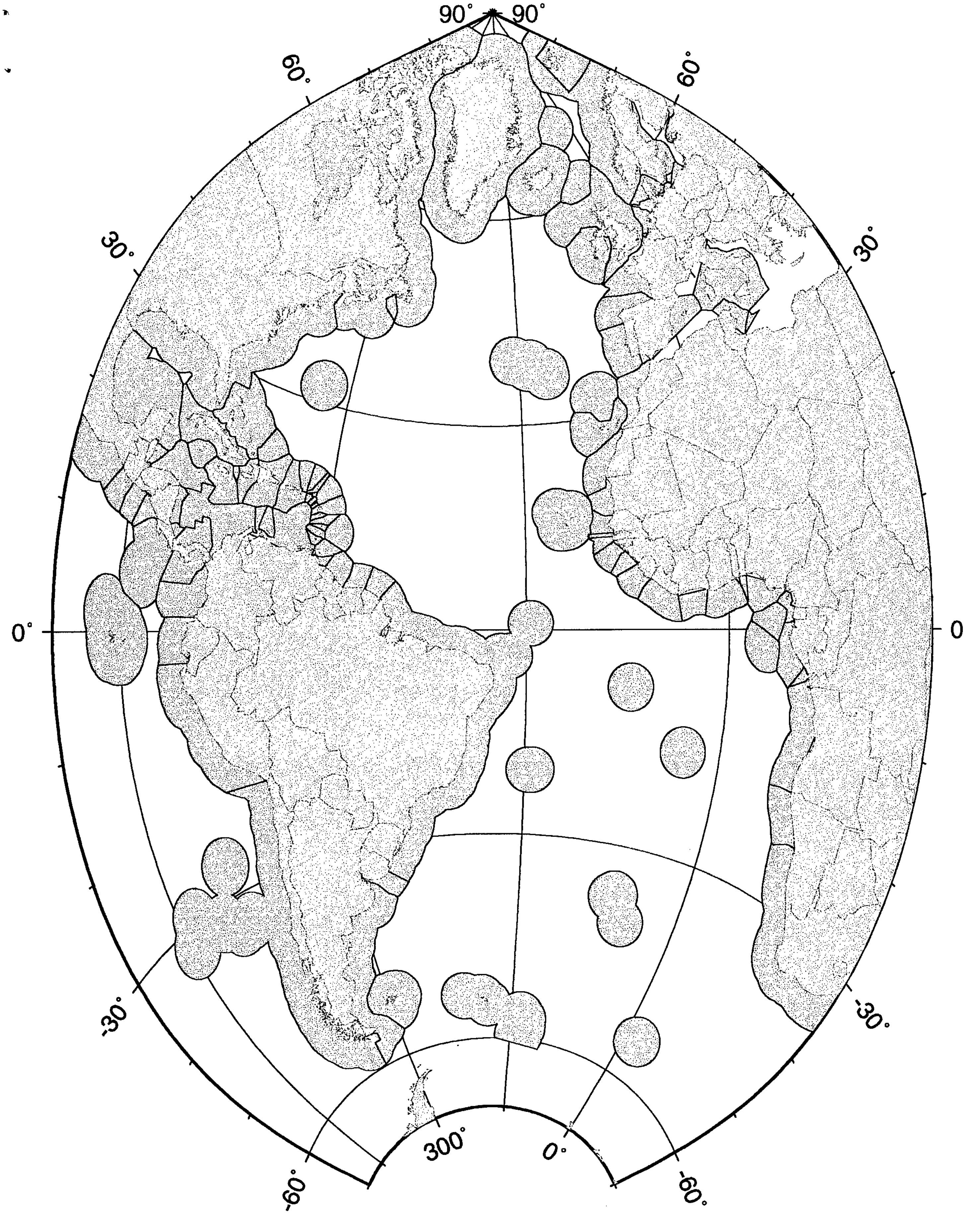
Appendix N: Summary of C language utility software

mj_clr.c	create line range files MJ_LRNG, MJ_LINED from ls****.seg files.
mj_car.c	extract line segments for countries and create file MJ_ARNG.
mj_cbr.c	create line range files MJ_BRNG, MJ_LINEB from bl****.bas files.
mj_init.c	performs the initialization of various paths and files.
mj_iac.c	initialize areas with country identifiers.
mj_ilr.c	initialize line segment ranges.
mj_ibr.c	initialize baseline ranges.
mj_iar.c	initialize rectangular range of an area, AREA_ID, ARANGE.
mj_iaa.c	initialize area attributes.
mj_getcg.c	extract a requested country identifier given a known property.
mj_gcn	get numeric area code given a two-character identifier.
mj_gc2	get two-character area code given a numeric area code.
mj_gc3	get three-character area code given a numeric area code.
mj_gct	get area text identifier given a numeric area code.
mj_gal.c	get set of line segment identifiers for an area (country).
mj_grl.c	get line segment identifiers overlapping a specified range.
mj_gapl.c	get ordered list of line segment identifiers for an area polygon.
mj_gar.c	get longitude and latitude extent of an area.
mj_gld.c	get coordinates for specified line segment identifier(s).
mj_gad.c	get coordinates for polygon border of specified area.
mj_gpra	get area identifiers for point in area range.
mj_gpca	get area identifiers for point in area.
mj_gab.c	get set of baseline identifiers for an area (country).
mj_gbd.c	get coordinates for baselines of specified area.
mj_pip.c	perform point in polygon test.
mj_incl.h	include file for applications

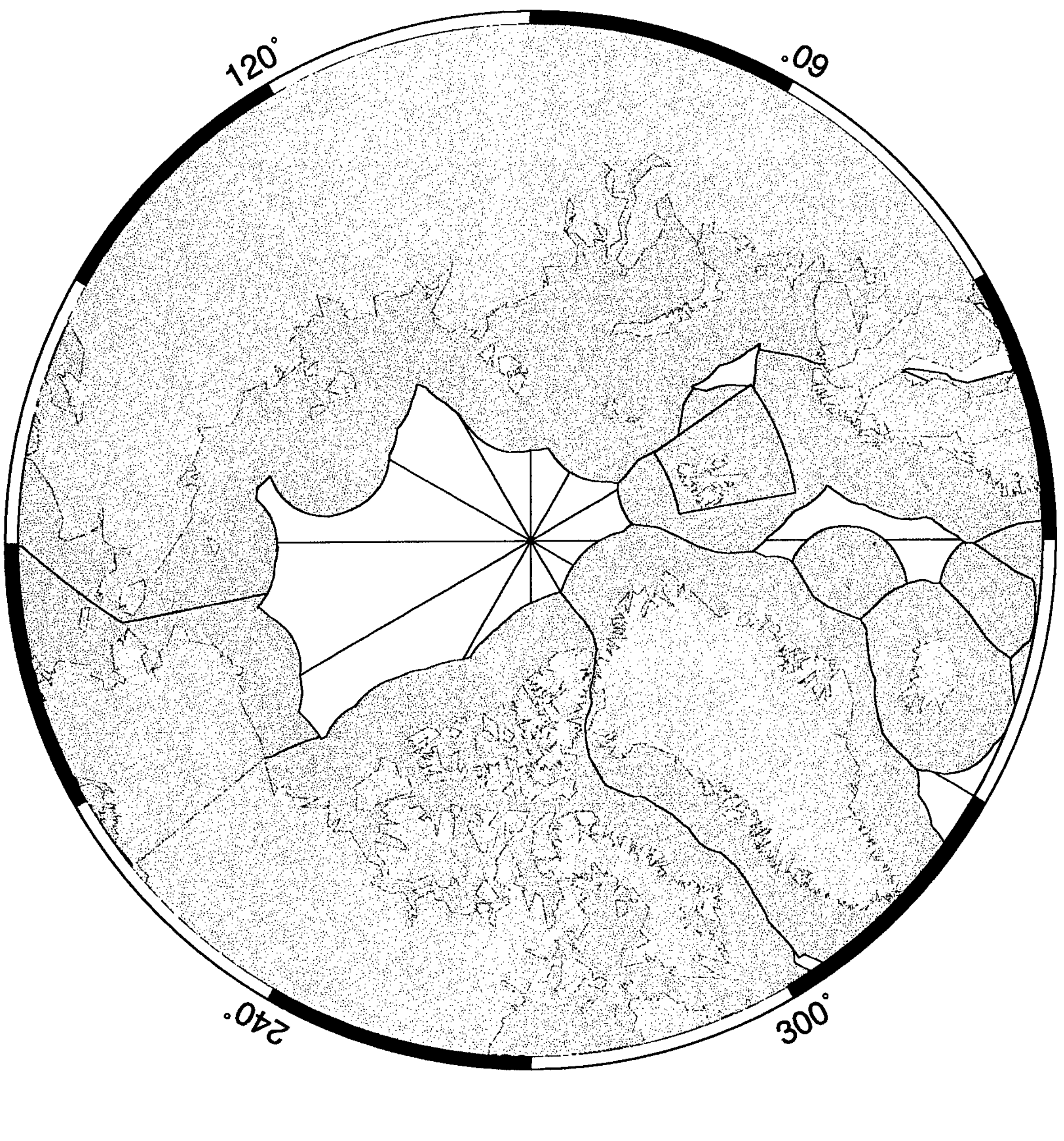
Appendix O: Summary of test programs

mjt_01.c	Test area/country identification code transformations. `mjt_01.exe method code_n code_a` For termination, enter: 0 0 code_a
mjt_02.c	Test line identifier retrieval by range. `mjt_02.exe lon_min lon_max lat_min lat_max`
mjt_03.c	Test line identifier retrieval by area code. `mjt_03.exe area_code_n`
mjt_04.c	Test area identifier (in range) retrieval by specified point. `mjt_04.exe lon_deg lat_deg`
mjt_05.c	Test line segment data retrieval. `mjt_05.exe seg_file_path seg_id`
mjt_06.c	Test line coordinate data retrieval by area code. `mjt_06.exe seg_file_path area_code_n`
mjt_07.c	Test area border data retrieval by area code. `mjt_07.exe area_code_n`
mjt_08.c	Test range retrieval by area code. `mjt_08.exe area_code_n`
mjt_09.c	Test area/country identifier (in polygon) retrieval by point. `mjt_09.exe lon_deg lat_deg`
mjt_10.c	Test baseline identifier retrieval by area code. `mjt_10.exe area_code_n`
mjt_11.c	Test attribute retrieval by area code. `mjt_11.exe area_code_n`


240°
300°
0°
60°
120°
180°
90°
90°
60°
60°
30°
30°
0°
0°
-30°
-30°
-60°
-60°
-90
-90°
240°
300°
0°
60°
120°
180°



90°
90°
60°
60°
30°
30°
0°
0
-30°
-30°
-60°
-60°
300°
0°


120°
60°
180°
0°
240°
300°

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE 3 March 1999	3. REPORT TYPE AND DATES COVERED Final 1/1/98 - 12/31/98

4. TITLE AND SUBTITLE Marine Jurisdictions Digital Database	5. FUNDING NUMBERS N00014-98-1-0276
6. AUTHOR(S) Roger A. Goldsmith	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Woods Hole Oceanographic Institution Woods Hole, MA 02543	8. PERFORMING ORGANIZATION REPORT NUMBER None
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, Code 321RF 800 North Quincy St. Arlington, VA 22217	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release Distribution Unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this project was to take the data gathered for the Maritime Claims chart and create a Maritime Jurisdictions digital database suitable for use with oceanographic mission planning objectives. To accomplish this a themed data set, associated tables, prototype processing software and documentation was developed. While GIS systems readily lend themselves to this task and the eventual application of the resulting data set, this product is generated as ASCII files with positions recorded in decimal degrees. This was done so as to maximize the deployment options. Users should be able to incorporate this data set into their own applications using a variety of hardware and operating systems. In this context it should also be noted this is not a deployable navigation and display application. The operating environments of the potential user community are simply too diverse. For display applications alone there are software packages such as Matlab, gmt, Surfer/MapViewer, ArcView, Atlas GIS, AutoCAD and many others, all driving a variety of CRTs, pen plotters and laser printers. To accommodate this diversity the product was made as a simple, easy to understand database that can be incorporated into a variety of applications.

14. SUBJECT TERMS Marine Jurisdictions Coastal State Boundaries	15. NUMBER OF PAGES
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL